รายงานวิจัยฉบับสมบูรณ์

โครงการพัฒนาหุ่นยนต์บินได้ควบคุมอัตโนมัติ

โดย รศ. ดร. มนูกิจ พานิชกุล

ธันวาคม 2548

สัญญาเลขที่ RSA/19/2545

รายงานวิจัยฉบับสมบูรณ์

โครงการพัฒนาหุ่นยนต์บินได้ควบคุมอัตโนมัติ

รศ. ดร. มนูกิจ พานิชกุล
สถาบันเทคโนโลยีแห่งเอเชีย

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย
(ความเห็นในรายงานนี้เป็นของผู้วิจัย สกว. ไม่จำเป็นต้องเห็นด้วยเสมอไป)

บทคัดย่อ

การควบคุมหุ่นยนต์บินให้บินอัตโนมัติเป็นเรื่องที่ค่อนข้างยาก นักวิจัยส่วนใหญ่จะประสพความสำเร็จในการควบคุมหุ่นยนต์ประเภทนี้เฉพาะในระบบจำลองเท่านั้น มีส่วนน้อยที่จะสามารถนำมาพัฒนากับหุ่นยนต์บินจริงได้สำเร็จ ในการที่จะควบคุมให้หุ่นยนต์บินสามารถทำงานได้ในแบบอัตโนมัตินั้น จำเป็นจะต้องควบคุมทั้งมุมและตำแหน่งของหุ่นยนต์ ในโครงการวิจัยนี้นักวิจัยได้ทดลองควบคุมหุ่นยนต์บินโดยใช้วิธีการควบคุมต่าง ๆ ที่เคยมีมา แต่พบว่าประสิทธิภาพไม่เป็นที่พอใจ ดังนั้นนักวิจัยจึงได้วิจัยค้นคว้าหาวิธีการควบคุมใหม่ที่เหมาะสมกับหุ่นยนต์บินโดยเฉพาะ ผลจากการทดลองพบว่าประสิทธิภาพของการควบคุมหุ่นยนต์บินเป็นที่น่าพอใจหากมุมของหุ่นยนต์บินจะถูกควบคุมโดยใช้ Neuro-Fuzzy Control (NFC) ในขณะที่ตำแหน่งของหุ่นยนต์จะใช้วิธีการควบคุมแบบผสมผสานกัน โดยวิธีควบคุมดังกล่าวใช้ชื่อว่า “Hybrid Adaptive Neuro-Fuzzy Model Reference Control (Hybrid-ANFMRC) ” ระบบควบคุมแบบ Neuro-Fuzzy จะถูกออกแบบโดยใช้ข้อมูลจากการบินจริงของหุ่นยนต์ ในขณะที่การควบคุมแบบ Hybrid-ANFMRC จะเรียนรู้และปรับตัวเองเพื่อให้ได้ลักษณะการตอบสนองตาม Reference Model ที่กำหนด ในขณะเดียวกันก็จะให้ผลการตอบสนองที่ดีและคา steady state error เป็นศูนย์ ประสิทธิภาพของระบบควบคุมที่ได้เสนอจะพิสูจน์โดยผลที่ได้จากการทดลอง

Abstract

Control of 6-DOF fully autonomous helicopter type flying robot is very difficult. Many researchers verified their control algorithms only on simulation. There are very few success experiments on fully control of the flying robot. In order to make the robot fly autonomously, the attitude and position controls are needed. In this research project, performance of the existing control algorithms was investigated. It was found that their performance was not satisfied to control fully autonomous flying robot. The researchers in this project propose new control algorithms which are specifically designed to control 6-DOF fully autonomous flying robot. Neuro-fuzzy controllers (NFC) are proposed to control roll, pitch and yaw angles of the flying robot, while the hybrid adaptive neuro-fuzzy model reference control (Hybrid-ANFMRC) is proposed to control the robot's position. The attitude controllers are trained offline to reduce roll, pitch and yaw errors. Position control in the flying robot applies a hybrid technique called, "hybrid adaptive neuro-fuzzy model reference control". The position controller learns online to track the velocity reference model, while trying to obtain smooth response and zero steady state error. Robustness design of the proposed control algorithm is addressed by testing in the experiments under various ranges of the controller gains. The experimental results show the satisfactory performance of the proposed control algorithm.

Executive Summary

ความสำคัญและที่มาของปัญหา

โดยธรรมชาติของสิ่งแวดล้อมที่หุ่นยนต์จะถูกนำไปใช้ เราสามารถแบ่งหุ่นยนต์เคลื่อนที่ออกเป็น 3 กลุ่มใหญ่ๆ คือ หุ่นยนต์ที่ทำงานบนพื้นและผนัง หุ่นยนต์ที่ทำงานในน้ำ และหุ่นยนต์ที่ทำงานในอากาศรวมถึงในอวกาศ ข้อจำกัดของหุ่นยนต์ประเภทต่างๆอยู่ที่การเข้าถึงตำแหน่งต่างๆ จะถูกกำหนดโดยลักษณะของพื้นที่ที่หุ่นยนต์เคลื่นที่ตัวนั้นถูกนำไปใช้ สำหรับหุ่นยนต์ที่ทำงานในอากาศหรืออวกาศนั้น องศาความมีอิสระมีอยู่ถึง 6 มิติครบถ้วน โดย 3 มิติแรก เป็นองศาความมีอิสระของตำแหน่งของหุ่นยนต์ ซึ่งประกอบไปด้วย ตำแหน่ง x , y และ z โดยเทียบกับแกนอ้างอิงบนโลกอีก 3 มิตหลังเป็นองศาความมีอิสระของทิศทางการหมุนของหุ่นยนต์ ซึ่งประกอบไปด้วยทิศทางการหมุนรอบแกน θ (roll), ϕ (pitch) , และ α (yaw) เนื่องจากองศาของความมีอิสระของหุ่นยนต์ที่ทำงานในอากาศ หรืออวกาศมีอยู่สูง ดังนั้นหุ่นยนต์บินได้จึงมีความสามารถในการเข้าถึงตำแหน่งต่างๆในทิศทางต่างๆในบริเวณการทำงานได้อย่างอิสระ อย่างไรก็ตามการควบคุมหุ่นยนต์บินได้ให้มีความสามารถในการเคลื่อนที่ได้โดยอัตโนมัตินั้น ยังเป็นสิ่งที่ทำได้ยากเนื่องจากมีปัจจัยหลายอย่างที่จะมามีผลต่อการเคลื่อนที่ของหุ่นยนต์ และหุ่นยนต์ยังต้องการอุปกรณ์ตรวจวัดประเภทต่างๆอีกหลายประเภท ที่มีความละเอียดแม่นยำสูงเพียงพอเพื่อใช้ในการควบคุมการเคลื่อนที่ของหุ่นยนต์

วัตถุประสงค์

1. เพื่อศึกษาและพัฒนาหุ่นยนต์บินได้ควบคุมอัตโนมัติขึ้นงานวิจัยนี้จะเป็นงานวิจัยที่บุกเบิก วงการหุ่นยนต์ในประเทศไทยให้มีความตื่นตัวและจะยังเป็นงานวิจัยที่สร้างศักยภาพของวงการหุ่นยนต์ในประเทศไทยให้ก้าวหน้าทัน หรือนำหน้าประเทศอื่นๆในโลกได้
2. เพื่อที่จะผลิตเทคโนโลยีใหม่ๆที่จำเป็นและเกิดขึ้นระหว่างกระบวนการพัฒนาหุ่นยนต์ บินได้ควบคุมอัตโนมัติ เช่นเทคโนโลยีการหาตำแหน่ง ทิศทางและความสูงของวัตถุที่เคลื่อนที่ ในบริเวณ 3 มิติเทคโนโลยีการควบคุมการเคลื่อนทีของวัตถุในอากาศ เทคโนโลยีการเรียนรู้แบ่งแยกสิ่งแวดล้อมภายนอก เทคโนโลยีการวางแผนการเคลื่อนที่และเทคโนโลยีอื่นๆอีกมากมาย เทคโนโลยีเหล่านี้สามารถถูกนำไปใช้ในสาขาอื่นๆ รวมถึงการนำไปประยุกต์ใช้ในโรงงานอุตสาหกรรมในอนาคตอีกด้วย
3. เพื่อที่จะได้นำหุ่นยนต์บินได้ควบคุมอัตโนมัตที่ได้รับการพัฒนาขึ้นไปใช้ในการศึกษาค้นคว้าวิจัยและพัฒนาต่อไปในอนาคต
4. ประโยชน์อื่นๆที่คาดว่าจะได้รับจากการวิจัย
    4.1 การนำหุ่นยนต์ไปใช้ในกิจกรรมทางการเกษตร เช่นการหว่านเมล็ดพืช ปุ๋ย หรือยากำจัดศัตรูพืชเป็นต้น
    4.2 การนำหุ่นยนต์ไปใช้ในการสำรวจทรัพยากรธรณี แหล่งน้ำ ป่า ประมงและอื่นๆ
    4.3 การนำหุ่นยนต์ไปใช้ในการติดต่อสื่อสารและการขนส่ง
    4.4 การนำหุ่นยนต์ไปใช้ในการบรรเทาและช่วยเหลือผู้ประสบภัยต่างๆ
    4.5 การนำหุ่นยนต์ไปใช้ในโรงงานอุตสาหกรรม
    4.6 การนำหุ่นยนต์ไปใช้ในสภาวะแวดล้อมที่มนุษย์ไม่เหมาะสมที่จะเข้าไปดำเนินการด้วยตัวเอง

ระเบียบวิธีวิจัย

ระเบียบวิธีการวิจัยประกอบด้วยขั้นตอนต่างๆดังนี้

ขั้นตอนที่1 ทำการศึกษาค้นคว้าและเลือกต้นแบบทีจะนำมาใช้เป็นหุ่นยนต์บินได้ควบคุมอัตโนมัติโดยพิจารณาจากหลักการทางเครื่องกล อากาศพลศาสตร์ ความสามารถในการรับภาระและสามารถในการแก้ไขดัดแปลงได้ของอุปกรณ์ต้นแบบ งานวิจัยนี้จะไม่เน้นสร้างหุ่นยนต์ต้นแบบใหม่ แต่จะเน้นเลือกต้นแบบที่มีอยู่แล้วแล้วนำมาดัดแปลงให้สามารถควบคุมได้โดยอัตโนมัติ

ขั้นตอนที่2 ทำการศึกษาค้นคว้าและพัฒนาเทคโนโลยีของการตรวจวัดต่างๆ ที่จำเป็นต่อการควบคุมหุ่นยนต์ เทคโนโลยีการตรวจหาตำแหน่งและทิศทางการหมุน เทคโนโลยีในการหาความสูง เทคโนโลยีในการแบ่งแยกสิ่งกีดขวางภายนอกและ เทคโนโลยีเสริมอื่น ๆ

ขั้นตอนที่ 3 ทำการศึกษาค้นคว้าและพัฒนาวิธีการควบคุมหุ่นยนต์ต้นแบบจากขั้นตอนที่ 1 วิธีการควบคุม (Control Algorithm) จะขึ้นอยู่กับส่วนขับเคลื่อนของตัวหุ่นยนต์ (Actuator) วิธีการควบคุมนี้จะต้องสามารถควบคุมหุ่นยนต์ให้เคลื่อนที่ได้อย่างสมดุลย์แม่นยำมีความคลาดเคลื่อนน้อยที่สุดและ การควบคุมต้องอัตโนมัติยืดหยุ่นและฉลาดต่อการเปลี่ยนแปลงของปัจจัยภายนอกที่เกิดขึ้นตลอดเวลาอีกด้วย

ขั้นตอนที่ 4 ทำการพัฒนาประกอบหุ่นยนต์บินได้ควบคุมอัตโนมัติให้เป็นจริงโดยประกอบอุปกรณ์ตรวจวัดต่าง ๆ เข้ากับหุ่นยนต์สร้างวงจรติดต่อสื่อสารเพื่อส่งข้อมูลต่างๆแล้วทำการโปรแกรมหุ่นยนต์ตามวิธีการควบคุมที่ได้พัฒนาขึ้นในขั้นตอนที่ 2

ขั้นตอนที่ 5 ทำการทดสอบ ประเมินผล และสรุป จากข้อมูลที่ได้จากการ Simulation และการทดลอง

1. บทนำ

ในอนาคตอันใกล้นี้ หุ่นยนต์บินจะถูกนำมาใช้งานแทนการทำงานของมนุษย์มากขึ้น โดยเฉพาะอย่างยิ่งในงานที่เสี่ยงต่ออันตราย ยกตัวอย่างเช่น การสอดแนมหรือถ่ายทอดภาพวีดีโอในบริเวณที่อาจเต็มไปด้วยสารเคมีที่เป็นอันตรายต่อมนุษย์ เป็นต้น ในการที่จะใช้งานหุ่นยนต์บินในลักษณะดังกล่าวได้ หุ่นยนต์บินจำเป็นต้องสามารถทำงานได้ในแบบอัตโนมัติ หุ่นยนต์บินที่พัฒนาขึ้นมาที่สถาบันเทคโนโลยีแห่งเอเชีย (AIT) ได้พัฒนาขึ้นมาให้สามารถบินในหลายๆลักษณะ นับต้องแต่การบินนิ่งอยู่กับที่ ไปจนกระทั่งการบินเคลื่อนที่ไปยังตำแหน่งต่างๆ ในโลกปัจจุบันได้มีนักวิจัยจำนวนมากที่กำลังพัฒนาหุ่นยนต์บินชนิดนี้ขึ้นมา มีเพียงส่วนน้อยเท่านั้นที่สามารถทำให้หุ่นยนต์บินสามารถบินได้ในแบบอัตโนมัติทั้งหมด เมื่อพิจารณาจะสามารถแบ่งกลุ่มนักวิจัยเหล่านี้ได้เป็น 2 กลุ่ม คือ นักวิจัยที่เน้นการพัฒนาโดยอาศัยแบบจำลองทางคณิตศาสตร์ของหุ่นยนต์บิน และนักวิจัยกลุ่มที่พัฒนาหุ่นยนต์บินโดยไม่ใช้แบบจำลอง นักวิจัยกลุ่มแรกส่วนมากจะสามารถควบคุมหุ่นยนต์บินได้ในระบบจำลองเท่านั้น ทั้งนี้เนื่องมาจากการออกแบบระบบควบคุมโดยใช้แบบจำลองทางคณิตศาสตร์ จำเป็นต้องหาแบบจำลองของหุ่นยนต์บินให้มีความแม่นยำพอ แต่หุ่นยนต์บินเป็นระบบที่ซับซ้อนมากๆ ทำให้การหาแบบจำลองดังกล่าวผิดพลาดไปจากความเป็นจริงมาก สุดท้ายจึงไม่สามารถนำมาพัฒนาเพื่อควบคุมหุ่นยนต์บินจริงได้ ดังนั้นในปัจจุบัน นักวิจัยอีกกลุ่มหนึ่งจึงได้มุ่งเน้นมาใช้วิธีการออกแบบโดยไม่ใช้แบบจำลอง ซึ่ง Fuzzy Logic และ Neural Network เป็นระบบควบคุมที่ถูกนำมาพัฒนาใช้กับหุ่นยนต์บินมากที่สุด

ในโครงการนี้ จะเป็นการพัฒนาระบบควบคุมการบินในลักษณะที่ไม่ใช้แบบจำลองทางคณิตศาสตร์ของหุ่นยนต์บิน ระบบควบคุมแยกออกเป็น 2 ระบบ ซึ่งใช้วิธีการควบคุมที่แตกต่างกัน การควบคุม มุม roll, มุม pitch และ มุม yaw ของหุ่นยนต์บินจะใช้วิธีการควบคุมโดยใช้ Neuro-Fuzzy Control ส่วนการควบคุมตำแหน่งของหุ่นยนต์บิน จะใช้วีการควบคุม Hybrid-ANFMRC การออกแบบระบบควบคุมโดยใช้ Neuro-Fuzzy Control จะไม่จำเป็นต้องใช้แบบจำลองทางคณิตศาสตร์ของหุ่นยนต์ การออกแบบจะอาศัยการสอนให้ระบบควบคุมสามารถควบคุมมุมของหุ่นยนต์บินได้ โดยใช้ข้อมูลที่บันทึกมาจากการบินของหุ่นยนต์บิน ส่วนการออกแบบระบบควบคุมตำแหน่งของหุ่นยนต์บินนอกจากไม่จำเป็นต้องใช้แบบจำลองทางคณิตศาสตร์แล้ว ระบบควบคุมยังออกแบบมาให้เรียนรู้และปรับตัวได้ด้วยตัวเองจนสามารถควบคุมให้หุ่นยนต์บินสามารถบินไปยังตำแหน่งต่างได้ตามที่ต้องการ ระบบควบคุมแบบ Hybrid-ANFMRC จะใช้เทคนิคการควบคุมแบบผสมผสานกันระหว่าง Proportional Control และ Adaptive Neuro-Fuzzy Model Reference Control โดยระบบควบคุมจะเรียนรู้ที่จะปรับตัวเองให้ได้การตอบสนองตามลักษณะของความเร็วการบินที่ต้องการ โดยรูปแบบของความเร็วดังกล่าว จะอยู่ในรูปของสมการที่มีความสัมพันธ์กับค่า position error ของหุ่นยนต์ขณะบิน สำหรับการควบคุมแบบ Neuro-Fuzzy จะใช้แบบจำลองทางคณิตศาสตร์ของหุ่นยนต์บินมาวิเคราะห์และประเมินประสิทธิภาพในการควบคุมก่อนที่จะมีการนำไปพัฒนาบนหุ่นยนต์บิน ซึ่งผลที่ได้จากการวิเคราะห์โดยใช้แบบจำลองทางคณิตศาสตร์และผลที่ได้จากการทดลอง ให้ผลการควบคุมที่ดีและคล้ายคลึงกัน ส่วนการควบคุมตำแหน่งจะใช้ผลที่ได้จากการทดลองมาประเมินประสิทธิการควบคุมเป็นหลัก รวมทั้งมีการทดสอบความคงทนของระบบควบคุมตำแหน่งโดยการเปลี่ยนค่า Proportional Gain ในช่วงต่างๆด้วยเช่นกัน

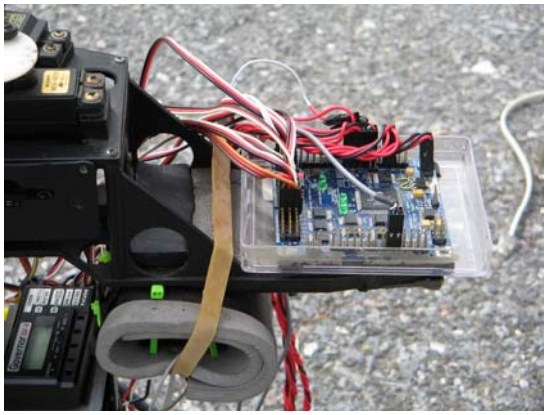
2. ระบบต่างๆของหุ่นยนต์บิน

หุ่นยนต์บินได้ถูกดัดแปลงมาจากเฮลิคอปเตอร์ขนาดเล็กบังคับด้วยวิทยุ มีเส้นผ่าศูนย์กลางของใบพัดหลักเท่ากับ 1.80 เมตร ติดตั้งเครื่องยนต์ที่มีกำลัง 3.0 แรงม้า ทำให้สามารถบรรทุกน้ำหนักได้ประมาณ 5.0 กิโลกรัม และบินได้นานประมาณ 15 นาที ในรูปที่ 1 แสดงให้เห็นหุ่นยนต์บินที่ติดต้องอุปกรณ์ที่ใช้ในการควบคุมเรียบร้อยแล้ว อุปกรณ์ที่ใช้ในการควบคุมการทำงานของหุ่นยนต์บินจะประกอบไปด้วยสิ่งต่างๆดังต่อไปนี้

- คอมพิวเตอร์ควบคุมการบิน ใช้คอมพิวเตอร์ขนาดเล็ก เป็นคอมพิวเตอร์แบบ 16-bits ทำหน้าที่ในการคำนวณและสร้างสัญญาณควบคุมไปขับ Servomotors ที่ติดตั้งใช้งานบนหุ่นยนต์บิน คอมพิวเตอร์ควบคุมการบินจะติดต่อกับคอมพิวเตอร์ควบคุมที่ภาคพื้นทุกๆ 0.2 วินาที เพื่อรับค่า DGPS และ คำสั่งต่างๆ รวมทั้งส่งค่าสถานะต่างๆของ การบินกลับลงไปที่ภาคพื้นด้วยเช่นกัน
- เซนเซอร์ วัดมุมของหุ่นยนต์บิน ภายในจะประกอบไปด้วย เซนเซอร์วัดอัตราเร็วเชิงมุม, เซนเซอร์วัดความเร่ง, เซนเซอร์วัดสนามแม่เหล็ก ครบทั้ง 3 แกน ข้อมูลที่ใช้จากเซนเซอร์ชนิดนี้คือ มุม Roll, มุม Pitch และ มุม Yaw ของหุ่นยนต์บิน
- จีพีเอส (GPS) จะให้ข้อมูลตำแหน่งของหุ่นยนต์บิน ด้วยความแม่นยำ 20 เซนติเมตร (CEP)
- เซนเซอร์วัดระยะสูง ประกอบไปด้วยเซนเซอร์ 2 ชนิด คือ Ultrasonic Sensor ใช้วัดระยะสูงจากพื้นดินเมื่อหุ่นยนต์บินในระยะต่ำ และ Pressure Altimeter วัดระยะสูงของหุ่นยนต์เทียบกับระดับน้ำทะเล ใช้ในกรณีที่หุ่นยนต์บินบินที่ความสูงมากๆ

รูปที่ 1 หุ่นยนต์บินอัตโนมัติ

นักวิจัยส่วนใหญ่ จะใช้คอมพิวเตอร์ PC104 เป็นคอมพิวเตอร์สำหรับการควบคุมการบิน แต่หุ่นยนต์บินที่พัฒนาขึ้นมานี้จะใช้คอมพิวเตอร์ 16-bits ในการประมวลผล ซึ่งมีข้อได้เปรียบในเรื่องของน้ำหนัก ขนาด และความประหยัดพลังงานไฟฟ้า ซึ่งจะเป็นประโยชน์อย่างมากสำหรับหุ่นยนต์บินที่มีข้อจำกัดในเรื่องของน้ำหนักและพื้นที่ที่ติดตั้งอุปกรณ์ซึ่งจำกัดมากๆ แต่ก็มีข้อจำกัดในเรื่องของความเร็วในการประมวลผล เนื่องจากการควบคุมหุ่นยนต์บินใน 1 วงรอบ จำเป็นต้องเสร็จสิ้นภายในเวลา 0.02 วินาที ฉะนั้นระบบควบคุมและวิธีการควบคุมที่เสนอนี้ นอกจากจะคำนึงถึงประสิทธิภาพในการควบคุมแล้ว ยังคำนึงถึงความเป็นไปได้ในการพัฒนาบนระบบคอมพิวเตอร์ขนาดเล็กนี้ด้วย ดังนั้น Membership Functions ที่ใช้ในบทความนี้จะเป็น Membership Function ที่เป็น สามเหลี่ยมสมมาตร เท่านั้น

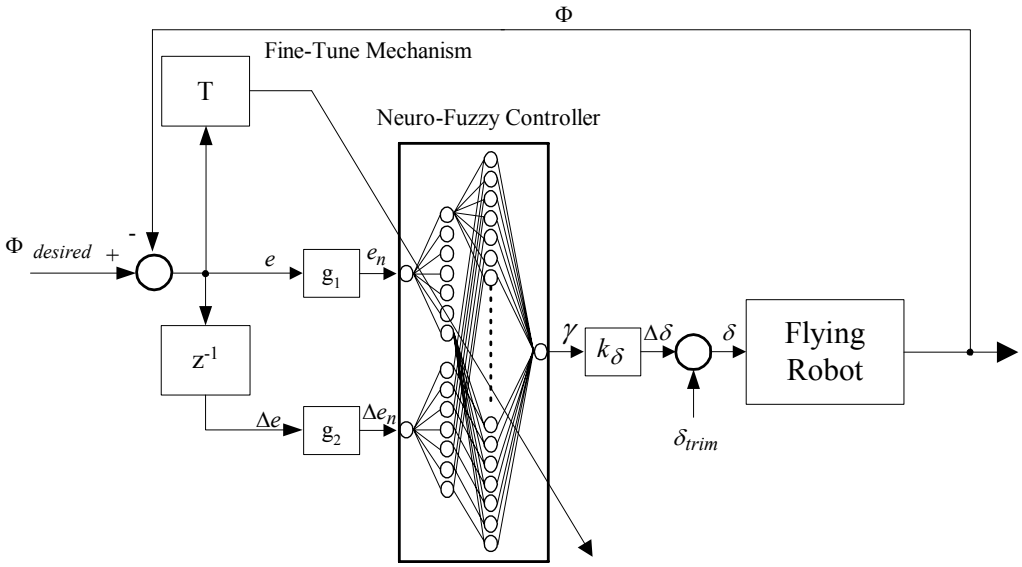


รูปที่ 2 คอมพิวเตอร์ควบคุมการบิน

3. ระบบการควบคุม

3.1 Neuro-Fuzzy Control

Neuro-fuzzy Control จะใช้ในการควบคุม มุม Roll, มุม Pitch และ มุม Yaw ของหุ่นยนต์บิน Neuro-Fuzzy Control เป็นระบบควบคุมที่ผสมผสานกันระหว่าง Fuzzy Logic และ Neural Network ระบบควบคุมที่ใช้แสดงไว้ใน รูปที่ 3



รูปที่ 3 ระบบควบคุมแบบ Neuro-fuzzy

ในรูปที่ 3 NFC มี 2 Inputs โดยที่ Input แรกคือ Attitude Error และ Input ที่สอง คือ Change of Attitude Error ส่วน Output คือ Change of Actuator Command และ Inputs ของ NFC คำนวณได้โดยใช้สมการที่ (1) และ สมการที่ (2) ตามลำดับ

$$e(k) = \Phi_{desired}(k) - \Phi(k) \quad (1)$$

$$\Delta e(k) = e(k) - e(k-1) \quad (2)$$

โดยที่ $\Phi_{desired}(k)$ คือ ค่า Desired Attitude และ $\Phi(k)$ คือ ค่ามุมของหุ่นยนต์บิน

Inputs ทั้งสองจะถูก Normalized ให้มีค่าอยู่ในช่วงที่ต้องการ การคำนวณแสดงได้โดยสมการที่ (3) และ สมการที่ (4) ตามลำดับ

$$e_n(k) = g_1(e(k)) \tag{3}$$

$$\Delta e_n(k) = g_2(\Delta e(k)) \tag{4}$$

โดยที่ $g_1(\bullet)$ และ $g_2(\bullet)$ คือ Function ที่ใช้ในการ Normalized ค่า Inputs ทั้งสองของ NFC ซึ่งแสดงได้โดยสมการที่ (5)

$$\begin{aligned} g_1(e(k)) &= k_e g_{1neg} e(k) && if \quad e(k) \le 0 \\ &= k_e g_{1pos} e(k) && if \quad e(k) > 0 \\ g_2(\Delta e(k)) &= k_{\Delta e} g_{2neg} \Delta e(k) \quad if \quad \Delta e(k) \le 0 \\ &= k_{\Delta e} g_{2pos} \Delta e(k) \quad if \quad \Delta e(k) > 0 \end{aligned} \tag{5}$$

โดยที่ k_e และ $k_{\Delta e}$ คือค่า Gains ของ Attitude Error และ Change of Attitude Error ตามลำดับ ส่วน g_{1neg}, g_{1pos}, g_{2neg} และ g_{2pos} เป็นค่าคงที่ที่ใช้ในการทำ Normalization

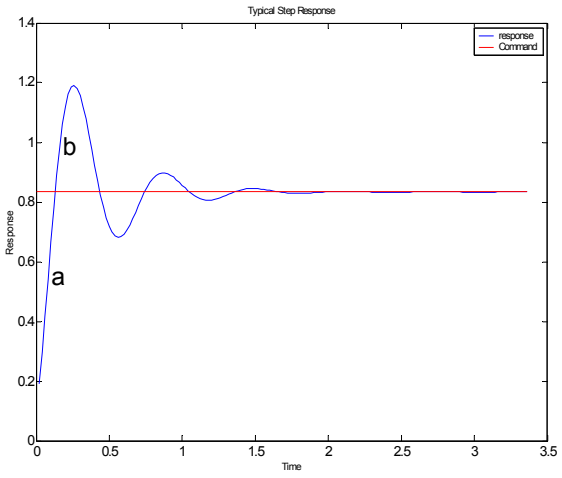
ด้วยค่า Normalized Attitude Error และ Normalized Change of Attitude Error จะสามารถคำนวณหาค่า Change of Actuator Command ได้ตามสมการที่ (6) และค่า Actuator Command จะคำนวณดังสมการที่ (7)

$$\Delta\delta(k) = k_\delta \gamma(k) \tag{6}$$

$$\delta(k) = \delta_{trim}(k) + \Delta\delta(k) \tag{7}$$

โดยที่ k_δ คือ ค่า Actuator Gain และ $\delta_{trim}(k)$ คือ ค่า Trim ของหุ่นยนต์บิน

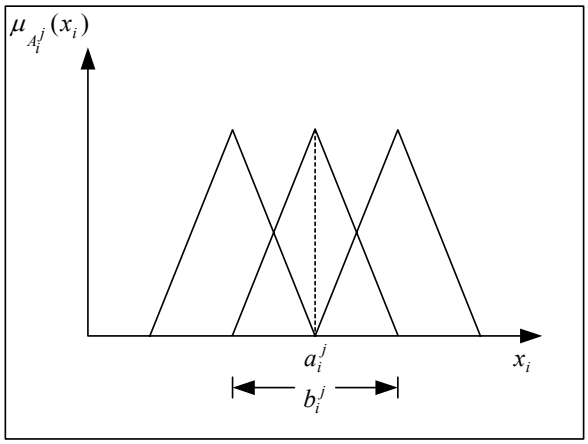
ในทางทฤษฎี สามารถปรับประสิทธิภาพในการควบคุมของ NFC ได้โดยการเปลี่ยนแปลงค่าคงที่ต่างๆที่ใช้ในการทำ Normalization ซึ่งในกรณีนี้ คือค่า Attitude Error Gain, ค่า Change of Attitude Error Gain และค่า Actuator Gain ในรูปที่ 4 แสดงให้เห็นถึงลักษณะทั่วไปของระบบที่ตอบสนองต่อค่า Command ที่เป็น Step Input



รูปที่ 4 Response ของระบบต่อ Step Input Command

พิจารณาบริเวณ “a” ในรูปที่ 4 จะเห็นว่าผลที่เกิดขึ้นนั้นเนื่องมากจากค่า Output มีค่าน้อย ซึ่งสามารถปรับปรุงระบบการควบคุมได้โดยการเพิ่มค่า Gain ของ Error และลดค่า Gain ของ Change of Error ซึ่งจะมีผลทำให้การควบคุมสามารถ Track ค่า Command ได้รวดเร็วขึ้น ส่วนในบริเวณ “b” ซึ่งเกิด Overshoots ขึ้น สามารถปรับปรุงระบบควบคุมได้โดยการเพิ่มค่า Gain ของ Change of Error ซึ่งจะสามรถลดการเกิด Overshoots หรือการเกิด Oscillations ลงได้ วิธีการดังกล่าวนี้จะนำมาใช้ในการปรับค่า Gains ของ NFC หลังจากผ่านขั้นตอนการ Train ระบบควบคุมด้วยข้อมูลจากการบินของหุ่นยนต์บินไปแล้ว

ในงานวิจัยนี้ NFC จะเริ่มต้นออกแบบด้วยการ Train แบบ Offline โดยใช้ข้อมูลซึ่งบันทึกมาจากการบินของหุ่นยนต์บิน โดยใช้วิธีการเรียนรู้แบบ Back Propagation Algorithm ในรูปที่ 5 แสดงรูปร่างของ Membership Functions ซึ่งในบทความจะใช้ Membership Functions ที่มีรูปทรงแบบสามเหลี่ยมสมมาตร



รูปที่ 5 Membership Functions ที่ใช้สำหรับ NFC

โดยที่ Membership Function ดังกล่าวสามารถแสดงได้ด้วยสมการที่ (8)

$$\mu_{A_i^j}(x_i) = 1 - \frac{2\left|x_i - a_i^j\right|}{b_i^j},\ i = 1,2,...n\ \ j = 1,2,...,m \tag{8}$$

เมื่อ x_i คือ ค่าของ Input(s), a_i^j คือ ค่ากึ่งกลางของสามเหลี่ยม และ b_i^j คือ ค่าความกว้างฐานของสามเหลี่ยม สำหรับกฎของ NFC [1] ที่ใช้คือ

Rule j: If x_i *is* A_1^j *and* x_2 *is* A_2^j and … and x_n is A_n^j *then* γ *is* w_j.

Output จาก NFC คำนวณโดยใช้สมการที่ (9)

$$\gamma(k) = \frac{\sum_{j=1}^{m} \mu_j(k) w_j(k)}{\sum_{j=1}^{m} \mu_j(k)} \tag{9}$$

โดยที่

$$\mu_j = \mu_{A_j^1}(x_i)\mu_{A_2^1}(x_i)\ldots \qquad \mu_{A_n^1}(x_i) \tag{10}$$

ในกระบวนการ Training ของระบบควบคุม Weights ของ NFC จะถูกปรับเปลี่ยนโดยการสร้าง Cost Function ขึ้นมา

ซึ่งนิยามได้ดังสมการที่ (11) ส่วนสมการที่ใช้ในการปรับค่า Weights ของ NFC แสดงไว้ในสมการที่ (12)

$$E = \frac{1}{2}(\Phi_{desired} - \Phi)^2 \tag{11}$$

$$w_j(k+1) = w_j(k) - \eta \frac{\partial E}{\partial w_j} \tag{12}$$

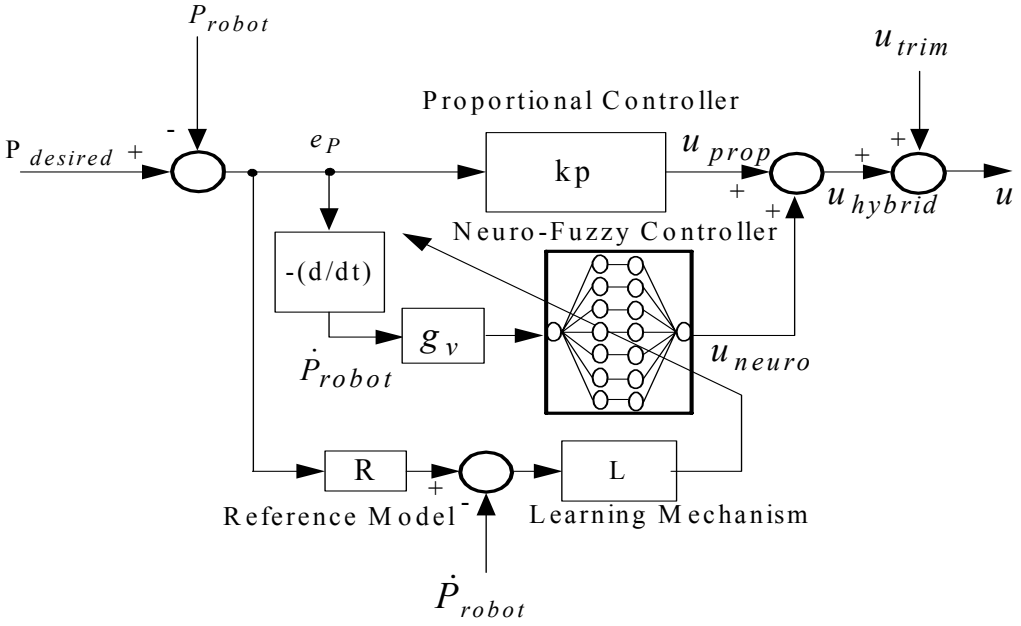
เมื่อ $\eta \geq 0$ คือ Learning Rate.

เทอม $\frac{\partial E}{\partial w^j}$ ในสมการที่ (12) สามารถหาได้จากการใช้ Chain Rule ดังแสดงในสมการที่ (13)

$$\frac{\partial E}{\partial w_j} = \frac{\mu_j(k)}{\sum_{j=1}^{m} \mu_j(k)} (\Phi_{desired}(k) - \Phi(k)) \tag{13}$$

3.2 Hybrid Adaptive Neuro-Fuzzy Model Reference Control (Hybrid-ANFMRC)

ในงานวิจัยนี้ Hybrid-ANFMRC จะใช้ในการควบคุมตำแหน่งของหุ่นยนต์บิน ระบบควบคุมดังกล่าวเป็นการนำเอาระบบควบคุมแบบ Proportional Control และระบบควบคุมแบบ Adaptive Neuro-Fuzzy Model Reference Control มาผสมผสานกัน โดยที่ Proportional Control จะสร้าง Output ที่มีค่าแปรผันตาม Error ที่เกิดขึ้น ในขณะที่ Adaptive Neuro-Fuzzy Model Reference Control จะสร้าง Output ออกมา โดยพยายามที่จะเรียนรู้และปรับตัวเองเพื่อให้ได้ผลการตอบสนองตาม Reference Model ที่กำหนดไว้ โครงสร้างของระบบควบคุมดังกล่าวแสดงในรูปที่ 6



รูปที่ 6 โครงสร้างของ Hybrid-ANFMRC

ในรูปที่ 6, ค่า Position Error คำนวณจากผลต่างระหว่าง Desired Position และ Position ของหุ่นยนต์บิน ดังแสดงในสมการที่ (14)

$$e_p(k) = P_{desired}(k) - P_{robot}(k) \tag{14}$$

ค่า Output จาก Proportional Control คำนวณได้ด้วยสมการที่ (15)

$$u_{prop}(k) = u_{prop}(k-1) + k_P(e_p(k) - e_p(k-1)) \tag{15}$$

เมื่อ k_P คือ ค่า Proportional Gain.

โดยปกติ ผลจากการใช้ระบบควบคุมแบบ Proportional Control จะทำให้ Error ของระบบลดลง แต่ถึงอย่างไรก็ตาม ก็จะยังคงมี Steady State Error เกิดขึ้น ฉะนั้น Adaptive Neuro-Fuzzy Model Reference Control จะสร้าง Output ออกมารวมกับค่า Output ที่ได้จาก Proportional Control โดยเรียนรู้และปรับตัวเองเพื่อให้เกิดการตอบสนองต่อการควบคุมตามรูปแบบของ Reference Model ที่กำหนดไว้ ซึ่งนอกจากจะทำให้ Steady State Error ไม่เกิดขึ้นแล้ว ยังมีข้อดีตรงที่สามารถกำหนดลักษณะของการตอบสนองของระบบที่ต้องการควบคุมได้ Outputs จากทั้งสองส่วนย่อยจะถูกนำมาคำนวณได้ดังแสดงด้วยสมการที่ (16) และค่า Output ของ Hybrid-ANFMRC คำนวณด้วยสมการที่ (17)

$$u_{hybrid}(k) = u_{prop}(k) + u_{neuro}(k) \tag{16}$$

$$u(k) = u_{hybrid}(k) + u_{trim} \tag{17}$$

Input ของ Adaptive Neuro-Fuzzy Model Reference Control คือ ค่าความเร็วของหุ่นยนต์ในแกนนั้นๆ โดยที่ค่าความเร็วดังกล่าวจะถูก Normalized ให้อยู่ในช่วงที่ต้องการ การ Normalization ดังกล่าวคำนวณโดยใช้สมการที่ (18)

$$\dot{P}_{n,robot}(k) = g_v \dot{P}_{robot}(k) \tag{18}$$

โดยที่ g_v คือค่า Normalized Factor

ส่วนค่า Output จาก Adaptive Neuro-Fuzzy Model Reference Control สามารถคำนวณได้ด้วยสมการที่ (19)

$$u_{neuro}(k) = \frac{\sum_{j=1}^{m} \mu_j(k) w_j(k)}{\sum_{j=1}^{m} \mu_j(k)} \tag{19}$$

โดยที่ $\mu_j(k)$ ในสมการที่ (18) คำนวณได้ดังแสดงในสมการที่ (20)

$$\mu_j(k) = A_1^i(\dot{P}_{n,robot}(k)) \tag{20}$$

เมื่อ $A_1^i(\dot{P}_{robot}(k))$ คำนวณได้เช่นเดียวกับสมการที่ (8)

Hybrid-ANFMRC จะเรียนรู้และปรับตัวเองให้ระบบมีการตอบสนองตามรูปแบบของ Reference Model ที่กำหนด โดยรูปแบบดังกล่าวจะอยู่ในรูปของ Function ของ Position Error ดังแสดงในสมการที่ (21)

$$r(k) = f(P_{robot}(k)) \tag{21}$$

โดยที่ $f(\bullet)$ คือ Function ที่เป็นได้ทั้งเชิงเส้น และไม่เป็นเชิงเส้น

Hybrid-ANFMRC จะปรับตัวเอง โดยใช้วิธีการปรับค่า Weights โดยการสร้าง Cost Function ขึ้นมา ซึ่ง Cost Function แสดงไว้ในสมการที่ (22)

$$E = \frac{1}{2}(r(k) - \dot{P}_{robot}(k))^2 \tag{22}$$

ส่วนสมการที่ใช้ในการปรับค่า Weights คือ

$$w_j(k+1) = w_j(k) - \eta \frac{\partial E}{\partial w_j} \tag{23}$$

โดยที่ $\eta \geq 0$ คือ ค่า Learning Rate.

ค่าของ $\frac{\partial E}{\partial w^j}$ สามารถคำนวณได้ด้วยสมการที่ (24)

$$\frac{\partial E}{\partial w_j} = \frac{\mu_j(k)}{\sum_{j=1}^{m} \mu_j(k)} (r(k) - \dot{P}_{robot}(k)) \tag{24}$$

4. ผลการทำ Simulations และผลการทดลองบิน

4.1 การจำลองการควบคุมมุม Yaw โดยใช้ NFC

การศึกษาโดยใช้แบบจำลอง เพื่อทดสอบประสิทธิภาพของ NFC ในการควบคุมมุมของหุ่นยนต์บิน ก่อนที่จะนำไปใช้จริงกับหุ่นยนต์บิน โดยใช้การควบคุมมุม Yaw ในการศึกษา แบบจำลองของหุ่นยนต์บินที่ใช้ได้มาจากบทความ J. Morris, M. van Nieuwstadt, and P. Bendotti. Identification and control of a model helicopter in hover. In Proceeding of the American Control Conference 1994. แบบจำลองดังกล่าวแสดงในสมการที่ (25) และสมการที่ (26)

$$x(k+1) = Ax(k) + Bu(k) \tag{25}$$

$$y(k) = Cx(k) \tag{26}$$

โดยที่

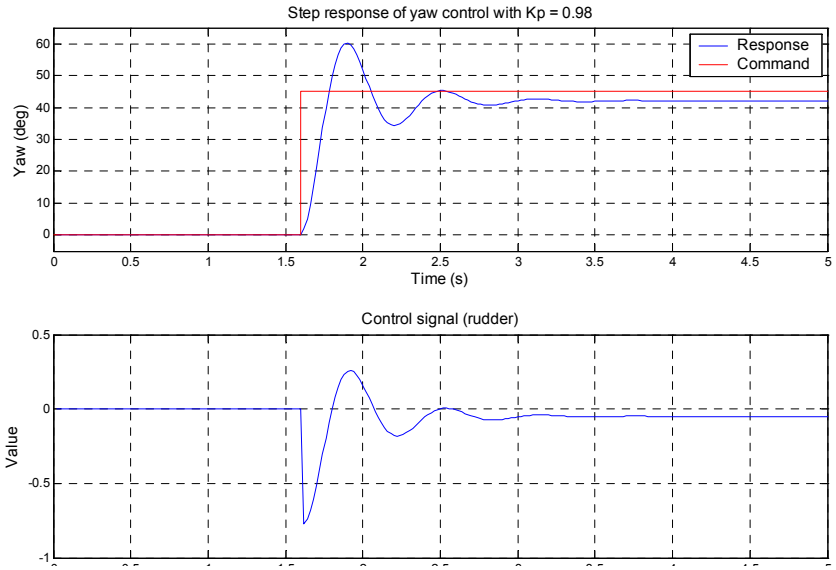
$$A = \begin{bmatrix} 1 & T_s \\ -0.1376 & 0.8947 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ -2.0269 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

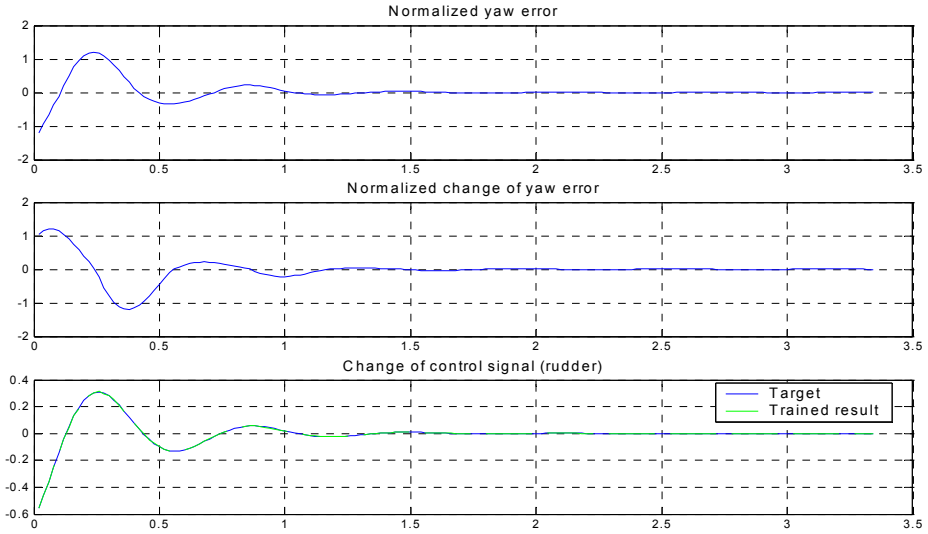
และ T_s คือ ค่า Sampling Time ซึ่งมีค่าเท่ากับ 0.02 วินาที.

ในการสร้างข้อมูลที่นำมาใช้ในการ Train ระบบควบคุมแบบ NFC การควบคุมแบบ Proportional Control จึงได้นำมาใช้ในการสร้าง Control Signals แทนค่าสัญญาณควบคุมจากคนบังคับหุ่นยนต์บิน โดยใช้วิธีการปรับค่า Proportional Gain จนกระทั่งระบบเกิดการ Oscillations ดังแสดงในรูปที่ 7 โดยค่า Desired Yaw คือ 45 องศา และค่า Proportional Gain คือ 0.98



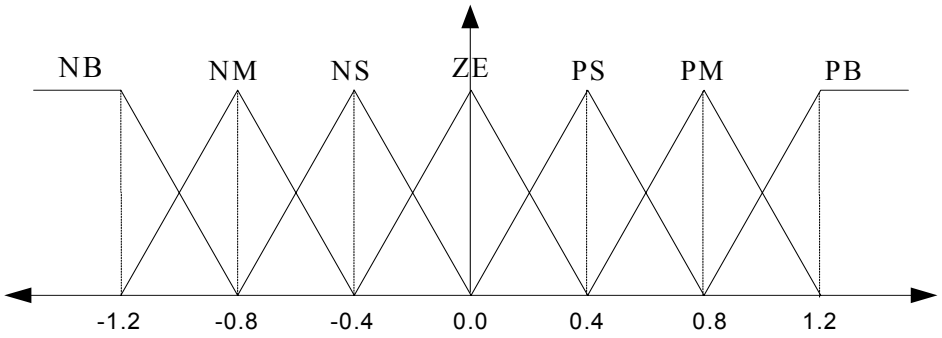
รูปที่ 7 Response จากการทำ Simulation ด้วยค่า Proportional Gain = 0.98

ในรูปที่ 7 เฉพาะข้อมูลบางส่วนเท่านั้นที่นำมาสร้างเป็นข้อมูลสำหรับใช้ในการเรียนรู้ของ NFC ข้อมูลดังกล่าวแสดงไว้ในรูปที่ 8



รูปที่ 8 ข้อมูลที่ใช้ในการเรียนรู้ของ NFC

ในการศึกษา NFC ออกแบบโดยใช้ 7-Membership Functions สำหรับ Inputs แต่ละตัว โดย Membership Function ดังกล่าวแสดงไว้ในรูปที่ 9 และค่าต่างๆที่ใช้ในการออกแบบ NFC แสดงไว้ในตารางที่ 1



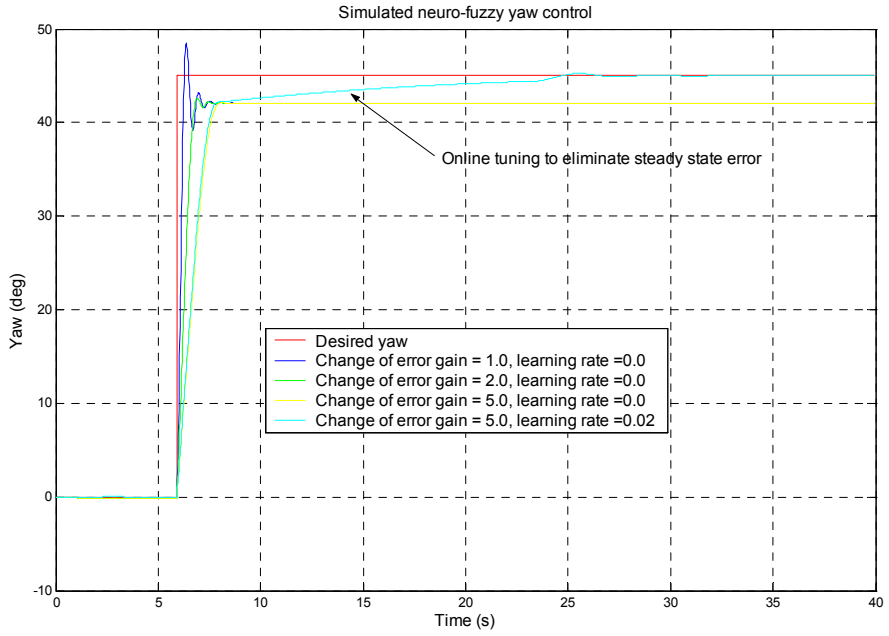
รูปที่ 9 Membership Function

ผลการศึกษาโดยใช้แบบจำลองแสดงในรูปที่ 10, ในช่วงเริ่มต้นก่อนที่จะมีการปรับค่า Gains ของ NFC จะสังเกตุเห็นการ Oscillations ของระบบ แต่ภายหลังการปรับค่า Gains แล้ว การควบคุมทำได้ดีขึ้น แต่ยังคงมี Steady State Error อยู่ในระบบ เพื่อขจัด Steady State Error การปรับแบบอัตโนมัติจึงมีความจำเป็น ภายหลังจากการใช้วิธีการปรับแบบอัตโนมัติดังกล่าว จะสังเกตุเห็นว่า Steady State Error จะไม่เกิดขึ้นในระบบ การปรับแบบอัตโนมัติจะใช้เกณฑ์ดังต่อไปนี้

$$\eta > 0.0 \qquad ,if \qquad |e(k)| \le a \ and \ |e(k)| \ge b \ and \ |\Delta e(k)| \le c$$
$$\eta = 0.0 \qquad ,otherwise$$

เมื่อ a , b และ c คือ ค่าคงที่ใดๆ ที่มีค่าเป็นบวก.

ในการศึกษา, ค่าของ Learning Rate คือ 0.02 ส่วนค่าของ a , b และ c คือ 4.0, 0.05 และ 0.1, ตามลำดับ ค่าคงที่ a และ c ใช้เพื่อป้องกันการปรับโครงสร้างของระบบควบคุมเมื่อระบบยังไม่เข้าสู่ Steady State ส่วนค่าของ b คือ ค่า Threshold ของการปรับ



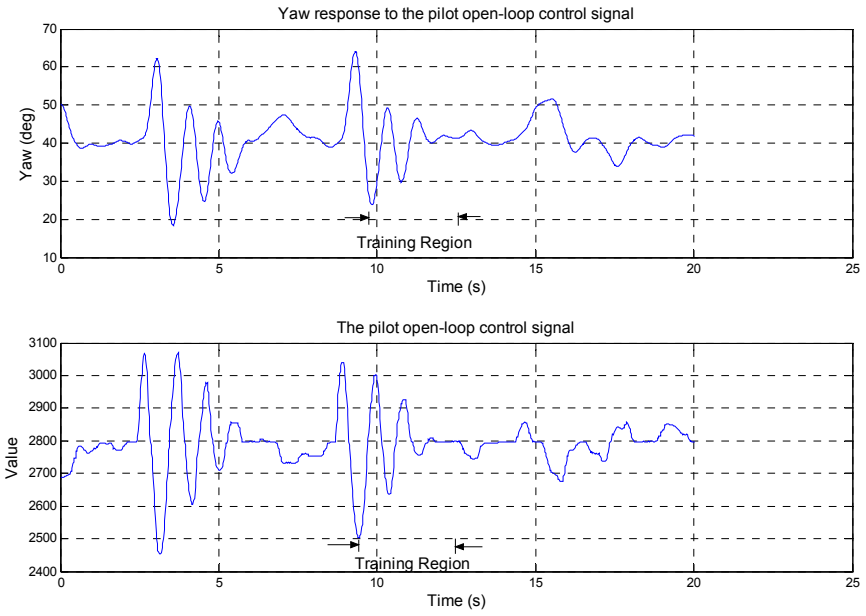
รูปที่ 9 ผลการควบคุมมุม Yaw โดยใช้ NFC

ตารางที่ 1 ค่าต่างๆที่ใช้สำหรับ neuro-fuzzy * แสดงค่าที่ได้หลังจากการปรับ

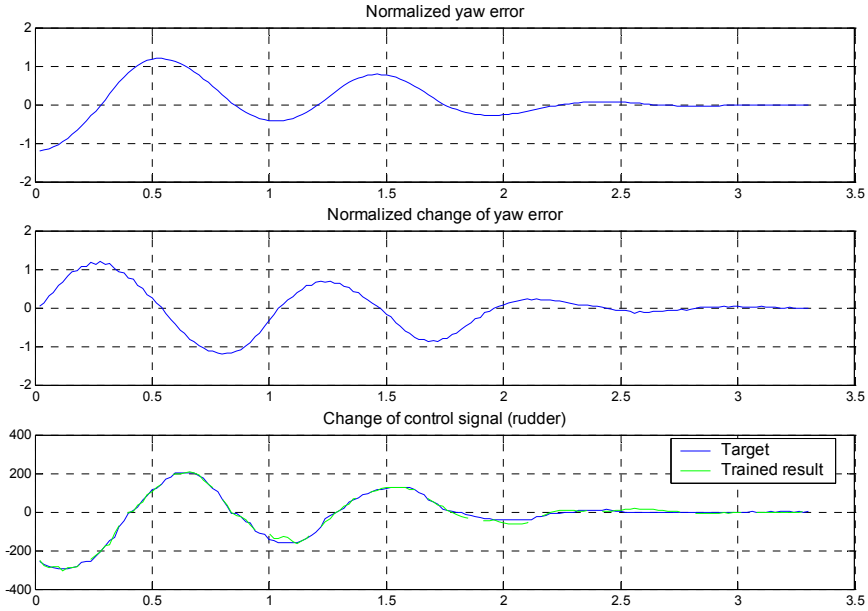
g_1		g_2		k		
g_{1neg}	g_{1pos}	g_{2neg}	g_{2pos}	k_e	$k_{\Delta e}$	k_δ
2.5453	3.8146	25.7061	10.9662	1.0 *1.0	1.0 *5.0	1.0 *1.0

4.2 ผลการทดลองควบคุมมุม Yaw โดยใช้ NFC

ในการทดลอง, ข้อมูลที่ใช้ในการเรียนรู้ของ NFC ได้มาจากการบินของหุ่นยนต์บิน โดยการเก็บข้อมูลดังกล่าวทำได้โดยการที่คนบังคับโยกคันบังคับบนชุดวิทยุควบคุมในลักษณะให้เกิดการแกว่งรอบจุดสมดุลของแกนการหมุนของมุม Yaw โดยมีการลดขนาดของการโยกลงจนกระทั่งไม่มีการหมุนรอบแกน Yaw ในขณะที่จะต้องพยายามให้แกนอื่นของหุ่นยนต์บินอยู่ในสมดุลในการบินให้มากที่สุด ข้อมูลที่บันทึกได้แสดงไว้ในรูปที่ 11



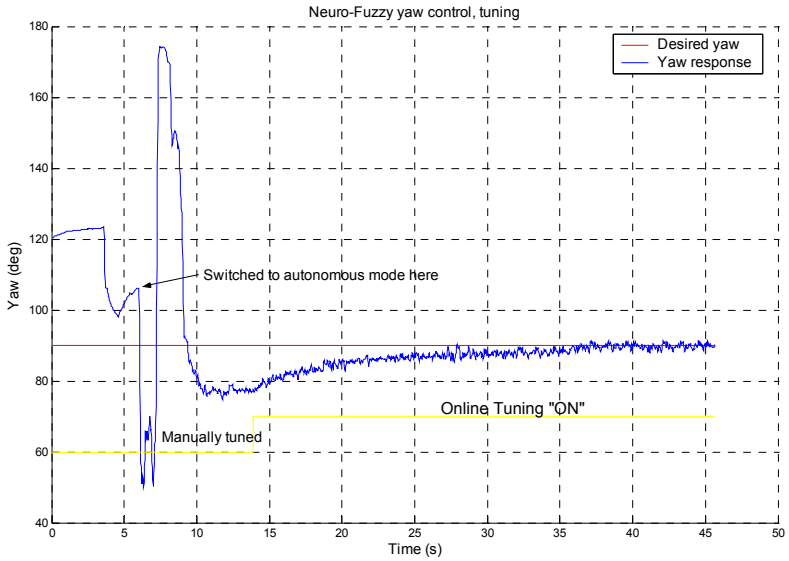
รูปที่ 11 ข้อมูลที่บันทึกจากการบินเพื่อนำมาใช้ในการเรียนรู้ของระบบควบคุม



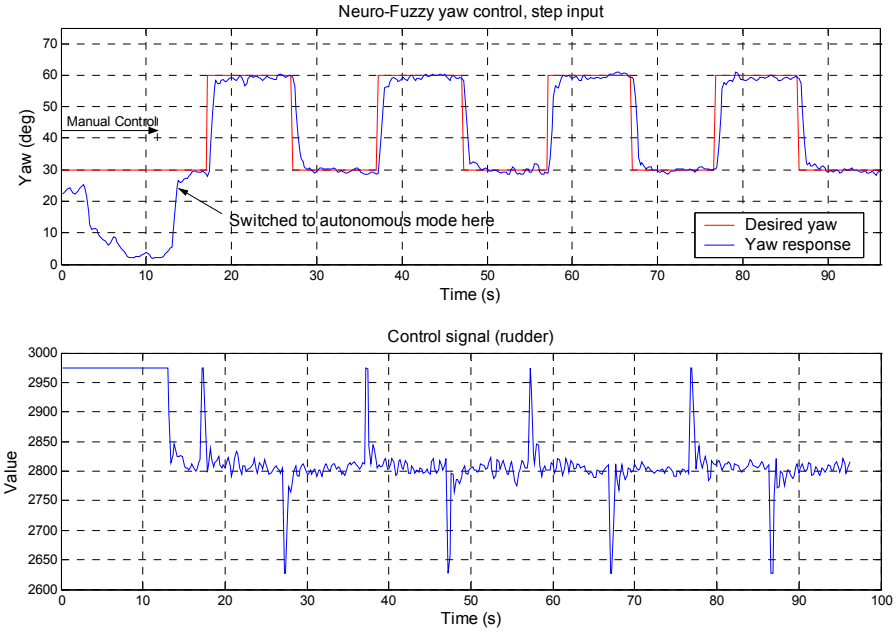
รูปที่ 12 ข้อมูลเพื่อใช้ในการเรียนรู้ของระบบควบคุม

ข้อมูลที่ใช้ในการเรียนรู้ของ NFC และผลจากเรียนรู้แสดงในรูปที่ 12 โดยในการทดลองจะใช้ Membership Functions ในรูปแบบเดียวกับในการศึกษาจากแบบจำลอง หลังจากการเรียนรู้ในขั้นตอนแรกเสร็จสิ้นลง ระบบควบคุมมีความจำเป็นที่จะต้องถูกปรับอีกครั้ง ซึ่งผลการทดลองดังกล่าวได้แสดงไว้ในรูปที่ 13 และทำนองเดียวกันกับการศึกษาด้วยแบบจำลอง ระบบยังคงมี Steady State Error ซึ่งสามารถขจัดได้ด้วยวิธีการปรับค่า Weights ของ NFC ในขณะบินเมื่อระบบเข้าสู่ Steady

State ผลดังกล่าวได้แสดงไว้ในรูปที่ 13 เช่นกัน ส่วนในรูปที่ 14 แสดงผลการทดลองเมื่อมีการเปลี่ยนค่า Desired Yaw ในลักษณะของ Step Input จะเห็นได้ว่า การใช้ NFC ตามที่ออกแบบมาจะให้ผลการควบคุมที่ไม่มี Overshoots หรือ การ Oscillations รวมทั้งค่า Steady State Error ยังมีค่าเป็นศูนย์อีกด้วย



รูปที่ 13 ผลการทดลองการควบคุมมุม Yaw ด้วย NFC



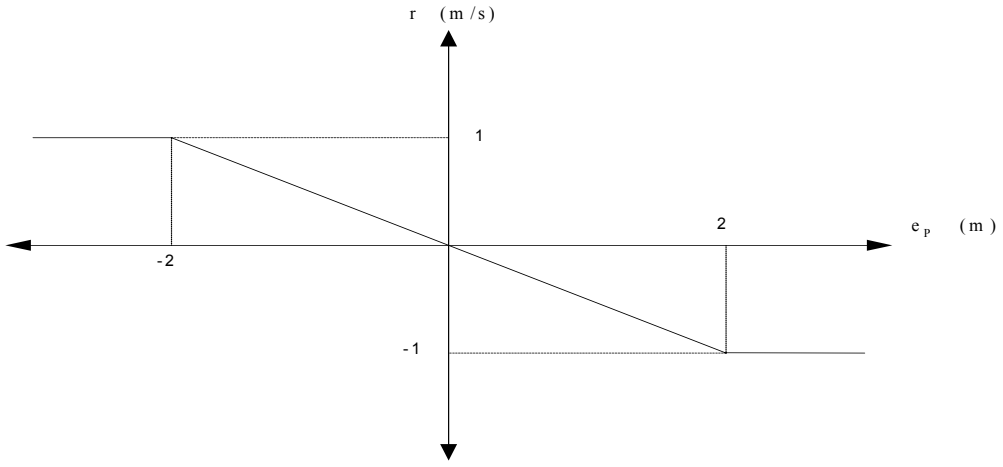
รูปที่ 14 ผลการทดลองเมื่อมีการเปลี่ยนค่า Desired Yaw

ตารางที่ 2 ค่าต่างๆที่ใช้สำหรับ neuro-fuzzy * แสดงค่าที่ได้หลังจากการปรับเปลี่ยน

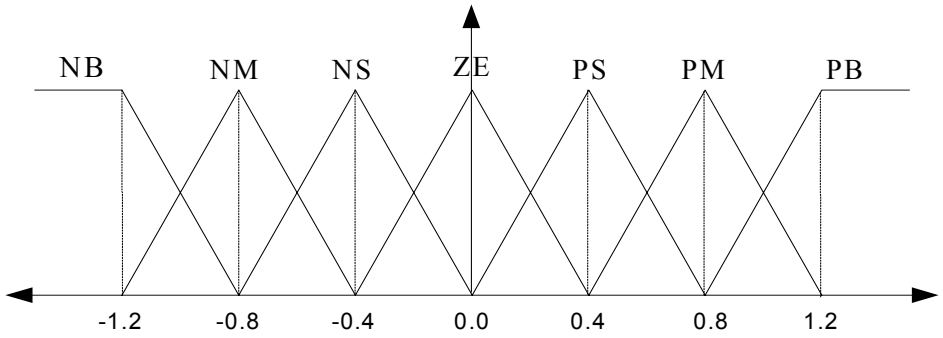
g_1		g_2		k		
g_{1neg}	g_{1pos}	g_{2neg}	g_{2pos}	k_e	$k_{\Delta e}$	k_δ
				1.0	1.0	1.0
0.0529	0.0684	0.7619	0.4706	*1.0	*2.1	*1.39

4.3 ผลการควบคุมตำแหน่ง โดยใช้ Hybrid-ANFMRC

รูปที่ 17, รูปที่ 18 และ รูปที่ 19 แสดงผลจากการใช้ Hybrid-ANFMRC ในการควบคุมตำแหน่งในแนว Lateral, Longitudinal และ ความสูง ตามลำดับ ค่า Outputs ของการควบคุมในแนว Lateral, Longitudinal และ ความสูง คือ ค่า Desired Roll, ค่า Desired Pitch และ ค่า Change of Collective Command, ตามลำดับ ส่วนการควบคุม มุม Roll, มุม Pitch ได้ใช้วิธีการควบคุมเช่นเดียวกับการควบคุม มุม Yaw ในบทความช่วงที่แล้วในการทดลองใช้ค่า Proportional Gains สำหรับแกน Lateral และ แกน Longitudinal เป็น 8.0 เท่ากัน ส่วนค่า Proportional Gain ของการควบคุมความสูงมีค่าเป็น 30.0 ค่า Desired Lateral Position และ Desired Longitudinal Position มีค่าเป็น 0.0 เมตร ค่า Desired Altitude คือ 13.0 เมตร ค่า Learning Rate ที่ใช้ในการควบคุมทั้งสามแกนมีค่าเท่ากันคือ 0.4 ส่วนVelocity Reference Model ที่ใช้ถูกกำหนดให้มีลักษณะของ Linear Function ดังแสดงในรูปที่ 15 Membership Functionsที่ใช้มีลักษณะเดียวกันกับที่ใช้ในการควบคุม มุม Yaw แตกต่างกันตรงที่ค่า Weights ของระบบควบคุมตำแหน่งที่จะมีเพียง 7 ค่าเท่านั้น ทั้งนี้เนื่องมาจากระบบควบคุมมี Input เพียงค่าเดียว นั้นคือ ค่าความเร็วของหุ่นยนต์บิน Membership Functions แสดงไว้ในรูปที่ 16.

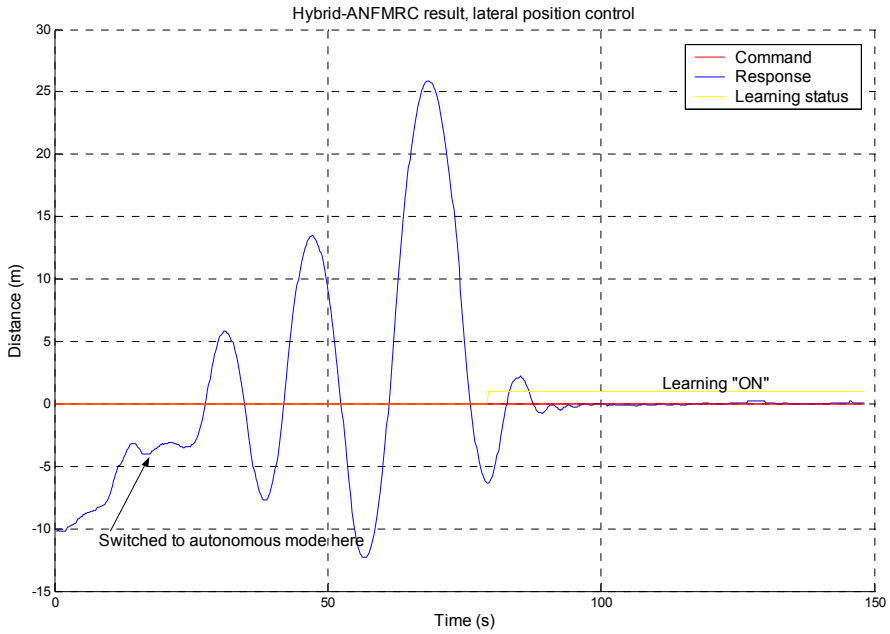


รูปที่ 15 Velocity Reference Model

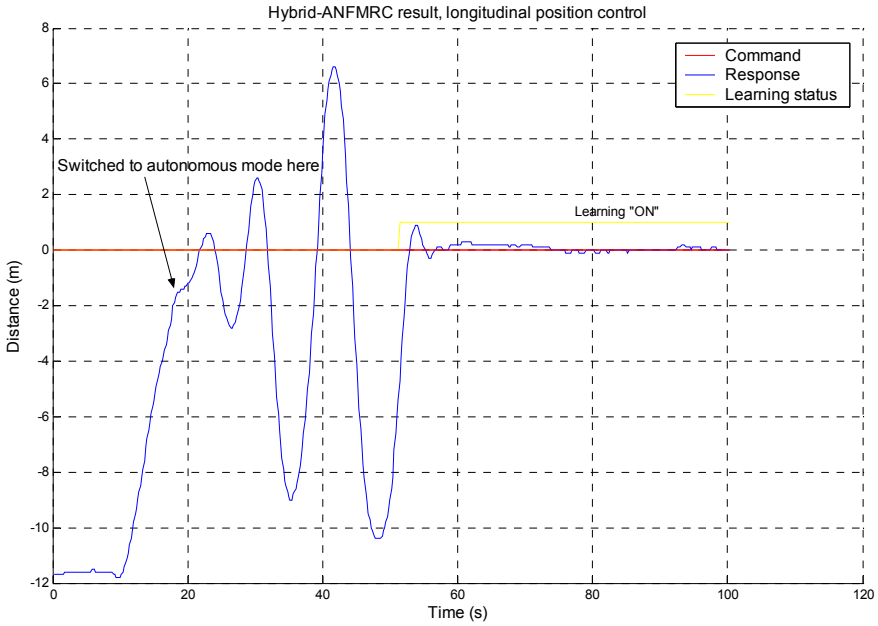


รูปที่ 16 Membership Functions สำหรับ Hybrid-ANFMRC

ในรูปที่ 17, รูปที่ 18 และ รูปที่ 19 เป็นผลการควบคุมตำแหน่งของหุ่นยนต์บิน โดยการทดลองเริ่มจากการใช้เพียง Proportional Control ในการควบคุมเพียงอย่างเดียวในช่วงเริ่มต้น หลังจากนั้นจึงได้มีการเริ่มต้นการเรียนรู้และปรับตัวเองของระบบควบคุม ซึ่งจากผลการควบคุมที่ได้ จะเห็นว่า ระบบควบคุมที่ออกแบบมามีประสิทธิภาพในการควบคุมตำแหน่งของหุ่นยนต์บินได้อย่างดี โดยที่ขณะเริ่มต้นการทดลองค่า Weights ทั้งหมดจะถูกกำหนดให้มีค่าเป็นศูนย์ นั่นหมายความว่า ระบบควบคุมจะต้องเริ่มต้นเรียนรู้ที่จะควบคุมตำแหน่งของหุ่นยนต์ให้ได้จากศูนย์

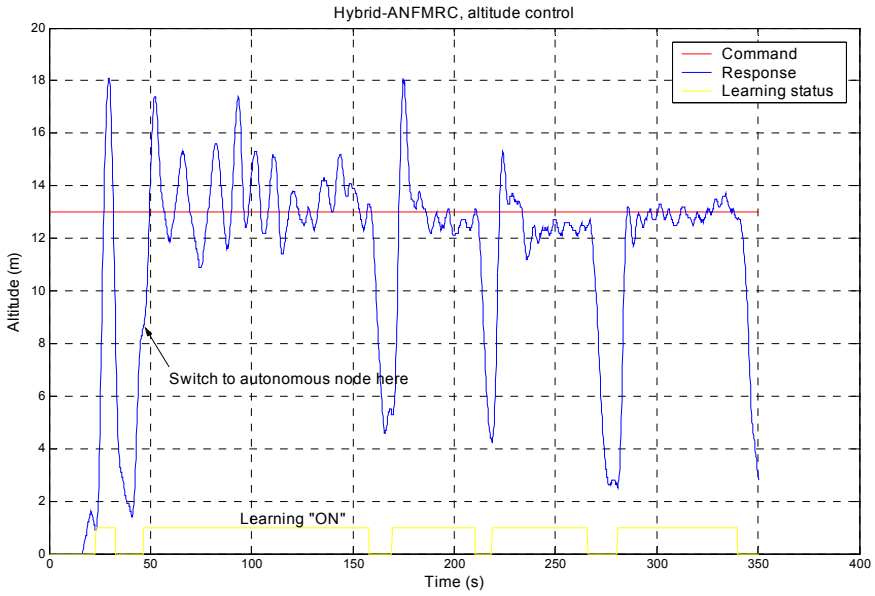


รูปที่ 17 Hybrid-ANFMRC, lateral position



รูปที่ 18. Hybrid-ANFMRC, longitudinal position

ในรูปที่ 19, เป็นผลการควบคุมความสูงของหุ่นยนต์บิน ระหว่างการบินจะมีการสลับการควบคุมระหว่างการควบคุมโดยอัตโนมัติและการควบคุมโดยคนบังคับ ขณะที่หุ่นยนต์ถูกควบคุมโดยคนบังคับ ระบบการเรียนรู้ของระบบควบคุมจะไม่ทำงาน และเมื่อกลับไปสู่การควบคุมโดยอัตโนมัติระบบการเรียนรู้จึงจะเริ่มต้นอีกครั้ง ทั้งนี้เพื่อเป็นการป้องกันไม่ให้ระบบควบคุมปรับตัวเองขณะที่หุ่นยนต์บินถูกควบคุมโดยคนบังคับ จากกราฟจะเห็นประสิทธิภาพของระบบควบคุมได้เป็นอย่างดี โดยในรอบแรกจะสังเกตุเห็นว่าเกิด Oscillations ขึ้น แต่ในรอบถัดๆไป Oscillations จะค่อยๆลดลง จนในที่สุดระบบควบคุมจะเรียนรู้ที่จะควบคุมความสูงของหุ่นยนต์บินได้อย่างดี



รูปที่ 19. Hybrid-ANFMRC, altitude

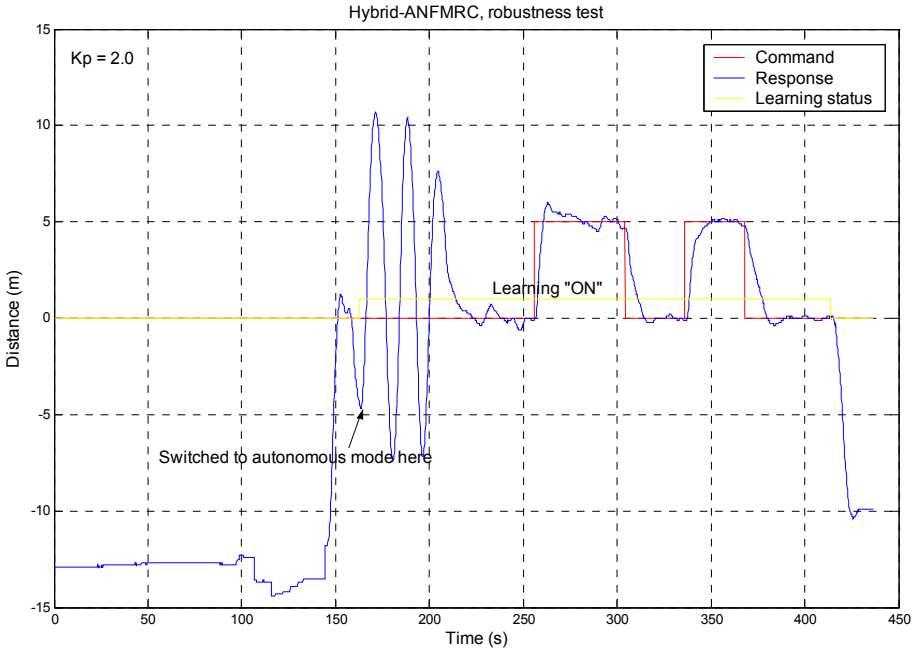
จากผลการทดลองข้างต้น, จะเห็นได้ว่า Hybrid-ANFMRC มีประสิทธิภาพในการควบคุมตำแหน่งของหุ่นยนต์บินได้เป็นอย่างดี ค่า weights ทั้งหมดของระบบควบคุมได้แสดงไว้ในตารางที่ 3

ตารางที่ 3. Weights of Hybrid-ANFMRC

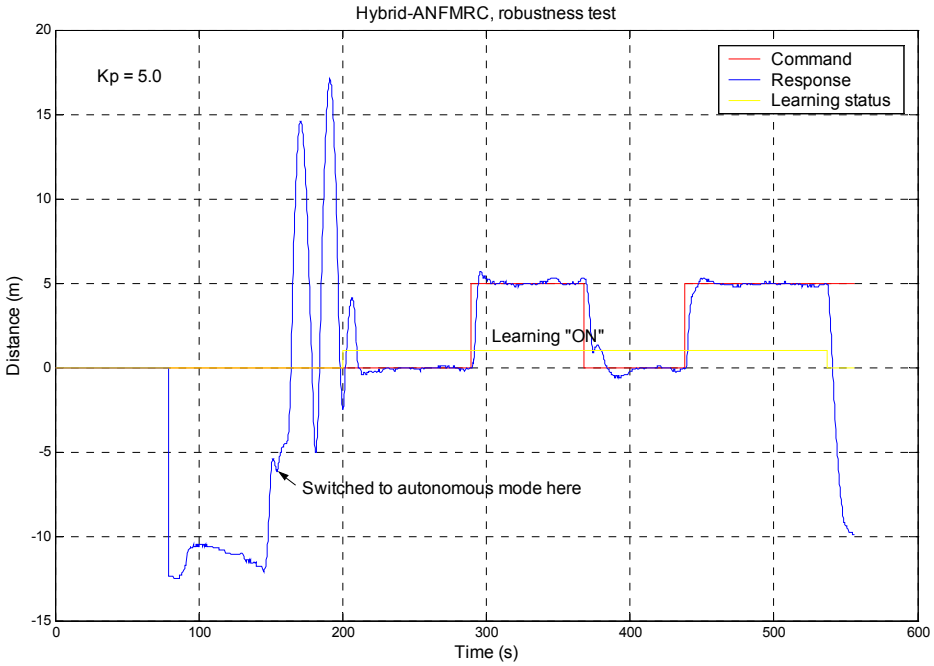
		w_1	w_2	w_3	w_4	w_5	w_6	w_7
Lateral	Before	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	After	104.9	13.65	10.67	-7.48	-14.31	-15.64	-58.63
Longitudinal	Before	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	After	-140.77	-15.8	-15.14	-2.18	8.35	13.61	83.79
Altitude	Before	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	After	-92.7	-23.8	-59.98	53.19	95.2	39.16	119.6

4.4 การทดสอบความ Robustness ของ Hybrid-ANFMRC

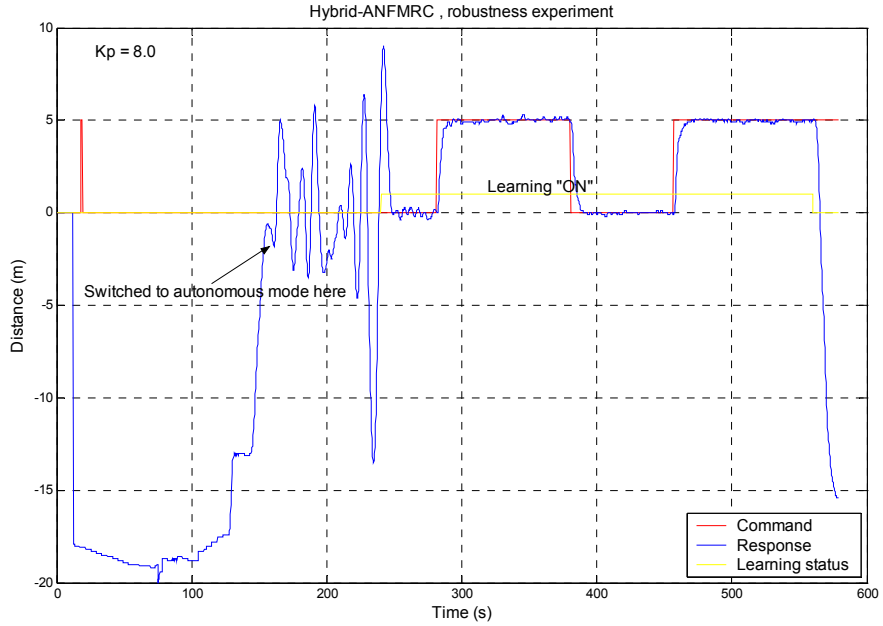
ในการทดลองนี้ เป็นการทดลองผลการควบคุมเมื่อมีการเปลี่ยนค่า Proportional Gain ในระบบควบคุม กรณีที่สามารถใช้ค่า Proportional Gain ได้ในช่วงกว้างนั้นจะมีข้อดีในแง่ของการออกแบบระบบควบคุมที่จะสามารถใช้ค่า Proportional Gain ได้โดยไม่ต้องใช้เวลามากนักในการปรับค่าดังกล่าว เพื่อจุดประสงค์นี้ การควบคุมตำแหน่งทางแกน Lateral จึงถูกนำมาศึกษาอีกครั้ง ซึ่งผลการควบคุมได้แสดงในรูปที่ 20, รูปที่ 21 และ รูปที่ 22 ด้วยค่า Proportional Gains เป็น 2.0, 4.0 และ 8.0 ตามลำดับ



รูปที่ 20 ผลการทดลอง Hybrid-ANFMRC ด้วย $k_P = 2.0$



รูปที่ 21 ผลการทดลอง Hybrid-ANFMRC ด้วย $k_P = 4.0$

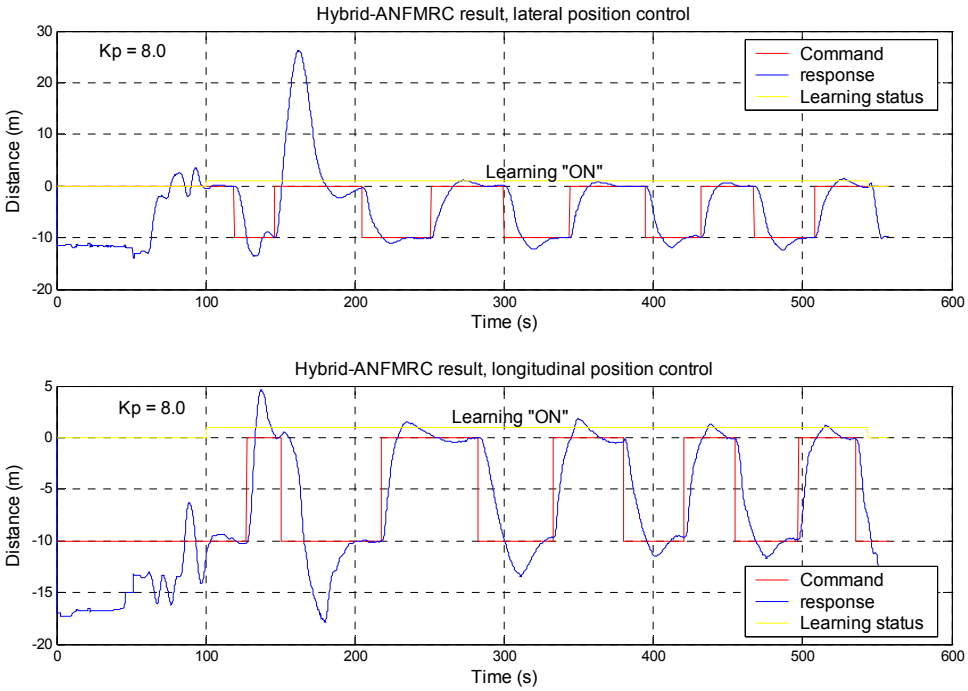


รูปที่ 22 ผลการทดลอง Hybrid-ANFMRC ด้วย k_P = 8.0

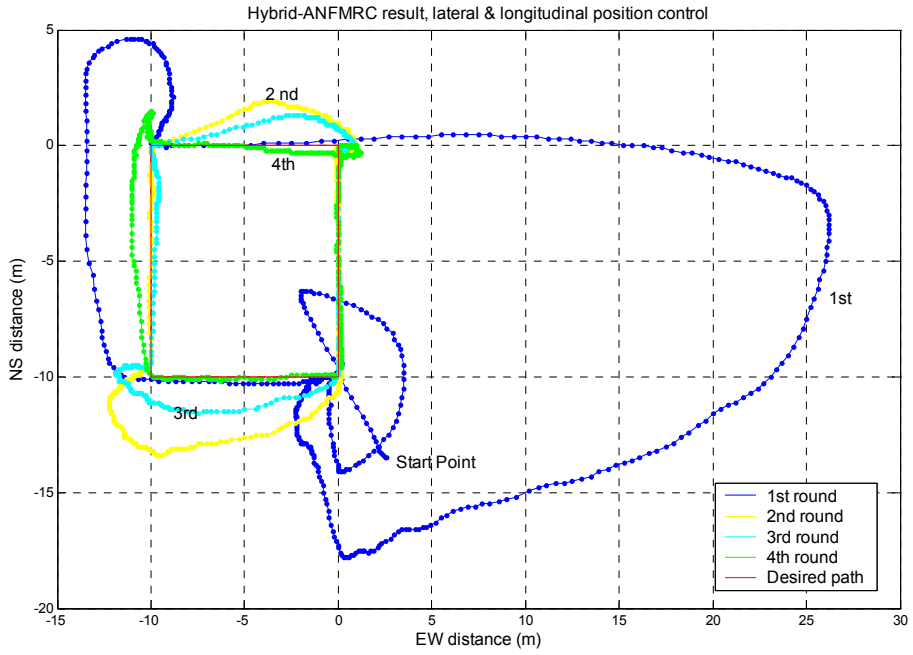
ในรูปด้านบน, เป็นการแสดงให้เห็นอย่างชัดเจนถึงความ Robust ของระบบควบคุมต่อการเปลี่ยนแปลงของค่า Proportional Gain โดยที่ระบบจะยังคงเรียนรู้ที่จะปรับตัวเองเพื่อให้ได้ผลของการควบคุมตามที่กำหนด อันเป็นการแสดงให้เห็นถึงผลของการใช้ระบบควบคุมชนิดที่สามารถปรับตัวเองได้

4.5 ผลการควบคุมแบบอัตโนมัติทั้ง 6 แกน

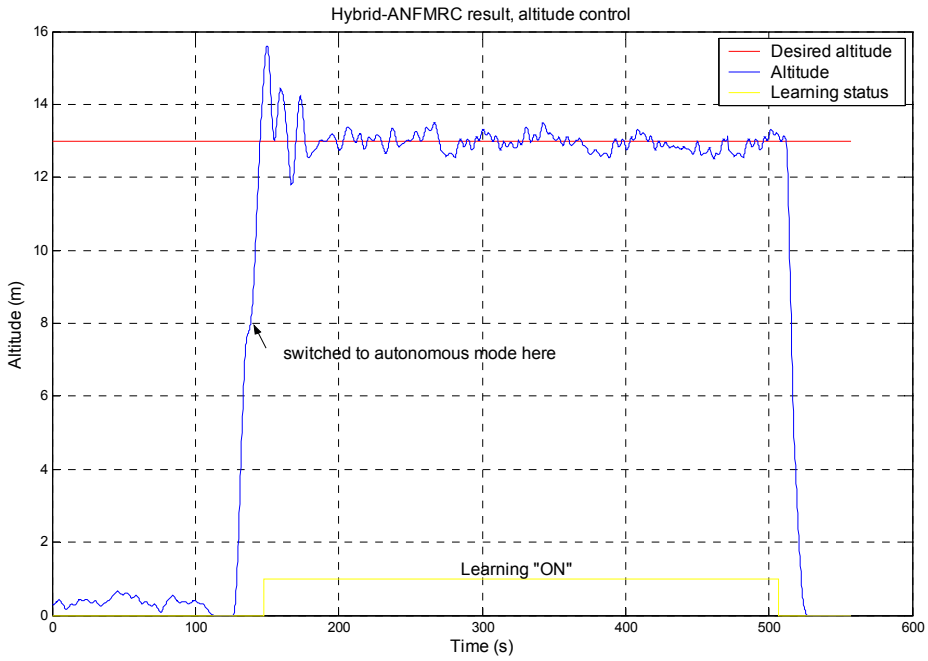
รูปที่ 23, รูปที่ 24, และ รูปที่ 25, แสดงผลการควบคุมตำแหน่งของหุ่นยนต์บิน เมื่อการควบคุมทั้งหมดเป็นแบบอัตโนมัติพร้อมกัน โดยที่หุ่นยนต์บินถูกกำหนดให้บินเคลื่อนที่เป็นสี่เหลี่ยมจัตุรัสขนาด 10x10 เมตร ค่า Desired Altitude คือ 13.0 เมตร ค่า Desired Yaw คือ 0.0 องศา เมื่อหุ่นยนต์บินบินเข้าไปในรัศมี 0.30 เมตรของแต่ละจุดจะถือว่าหุ่นยนต์บินเข้าไปยังจุดที่ต้องการ และบินไปยังจุดต่อๆไปในลักษณะนี้เรื่อยๆ



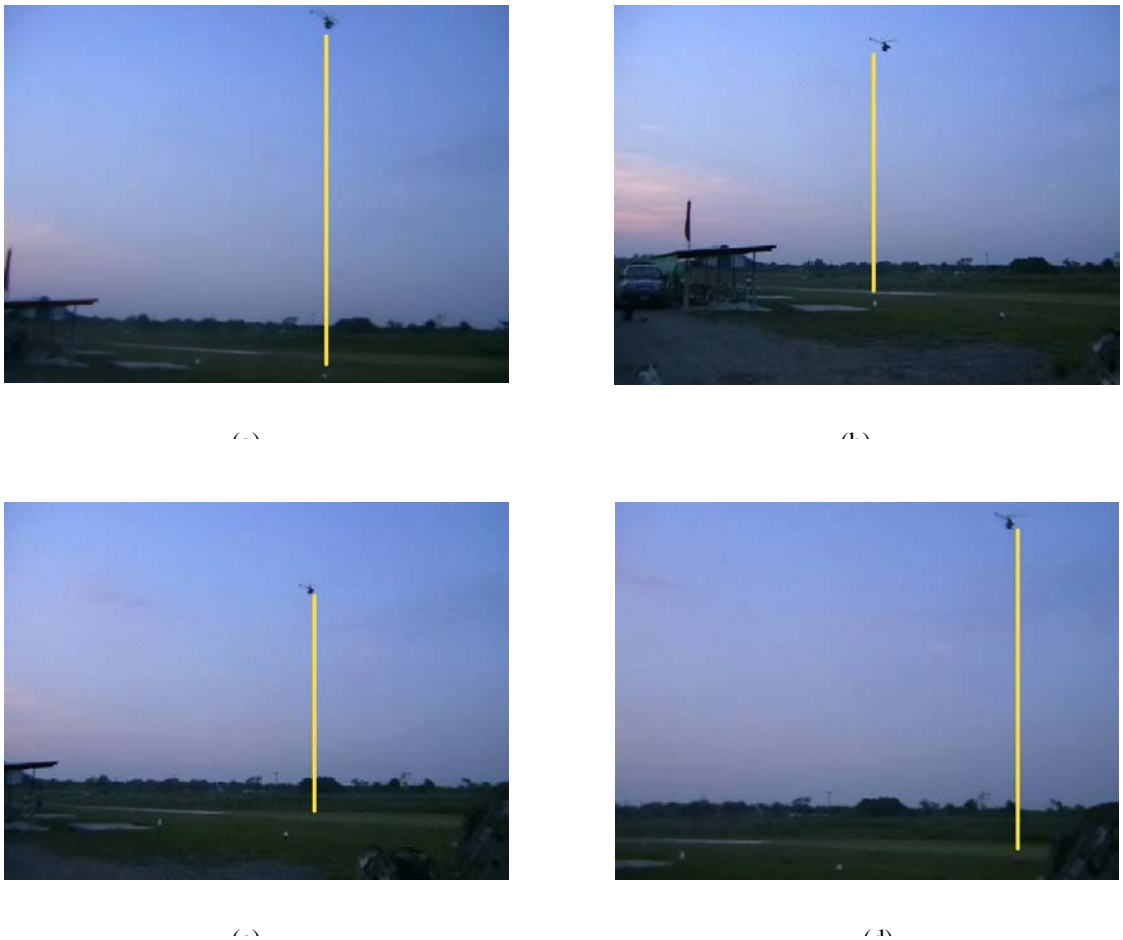
รูปที่ 23 ผลการควบคุมตำแหน่งของหุ่นยนต์บิน, ตำแหน่งทางแกน lateral และ longitudinal



รูปที่ 24 ผลการควบคุมตำแหน่งของหุ่นยนต์บิน



รูปที่ 25 ผลการควบคุมความสูง



รูปที่ 26 ภาพแสดงแบบอัตโนมัติ

(a) บินผ่านจุดที่ 1, (b) บินผ่านจุดที่ 2, (c) บินผ่านจุดที่ 3, (d) บินผ่านจุดที่ 4

จากผลการควบคุมจะเห็นว่าระบบควบคุมสามารถเรียนรู้และปรับตัวเอง จากค่า Weights ทั้งหมดมีค่าเป็นศูนย์ จนกระทั่งสามารถควบคุมตำแหน่งของหุ่นยนต์ให้สามารถบินไปยังตำแหน่งต่างๆที่ต้องการได้เป็นอย่างดี

5. สรุป

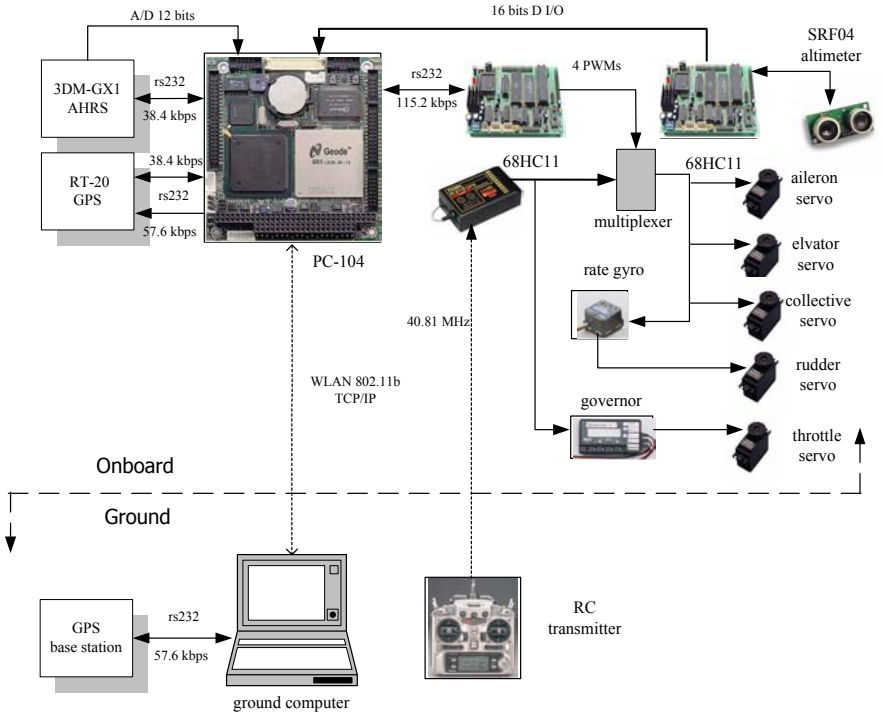
งานวิจัยนี้เป็นการนำเสนอและออกแบบระบบควบคุมสำหรับหุ่นยนต์บิน โดยใช้ Neuro-Fuzzy Control (NFC) ในการควบคุม มุม roll, มุม pitch และ มุม yaw ใช้ Hybrid-ANFMRC ในการควบคุมตำแหน่งของหุ่นยนต์บิน การออกแบบระบบควบคุมด้วย NFC เริ่มต้นด้วยการใช้ข้อมูลจากการบินของหุ่นยนต์บินมาทำการ Train ให้ได้โครงสร้างของระบบควบคุม หลังจากนั้นจึงทำการปรับระบบควบคุมโดยการใช้ควบคุมหุ่นยนต์จริงๆ และการเปิดการปรับตัวเองของระบบควบคุมเพื่อให้ Steady State Error หมดไป การออกแบบระบบควบคุมตำแหน่ง ซึ่งใช้ Hybrid-ANFMRC เริ่มต้นจากการปรับค่า Proportional Gain ซึ่งระบบควบคุมแบบนี้มีข้อดีคือ สามารถเลือกใช้ค่า Proportional Gain ได้ในช่วงที่กว้าง เนื่องจากระบบควบคุมเป็นระบบที่สามารถเรียนรู้และปรับตัวเองได้ โดยมีข้อแม้เพียงแต่ว่า อย่าทำให้เกิดการ Saturation ขึ้นในช่วงตำแหน่งที่ต้องการควบคุม หลังจากนั้นระบบจะเริ่มเรียนรู้และปรับตัวเองจนได้ประสิทธิภาพตามที่กำหนด จากผลของการควบคุมหุ่นยนต์บินให้บิน

อัตโนมัติ ได้แสดงให้เห็นถึงประสิทธิภาพในการควบคุมด้วยระบบควบคุมดังกล่าว ทั้งนี้ในบทความนี้ใช้เพียง Reference Model ที่เป็นเส้นตรง ซึ่งสามารถที่จะเปลี่ยนไปใช้ Reference Model ที่เป็นลักษณะที่ไม่เป็นเชิงเส้น เช่น Exponential Function ก็จะสามารถเพิ่มประสิทธิภาพของการควบคุมได้ดียิ่งขึ้น

ผลลัพธ์ที่ได้

1. ได้ต้นแบบของหุ่นยนต์บินได้ที่เหมาะสมที่สามารถนำมาใช้จริง ดังแสดงในรูปต่อไป

2. ได้อุปกรณ์ตรวจวัดต่าง ๆ ที่จำเป็นในการควบคุมหุ่นยนต์บินได้แบบอัตโนมัติ โดยมีการเชื่อมต่อของอุปกรณ์ต่าง ๆ ดังแสดงในแผนภูมิต่อไปนี้



3. ได้วิธีการควบคุมที่เหมาะสมที่สามารถควบคุมหุ่นยนต์บินได้ ในการควบคุม มุม roll, มุม pitch และ มุม yaw ของหุ่นยนต์บินจะใช้วิธีการควบคุมโดยใช้ Neuro-Fuzzy Control และในการควบคุมตำแหน่ง x, y, และ z ได้ใช้วิธีการควบคุมแบบ Hybrid-Adaptive Neuro-Fuzzy Model Reference Control

4. ได้หุ่นยนต์บินได้ควบคุมอัตโนมัติที่สมบูรณ์ วิดีโอเทปการบินแบบควบคุมอัตโนมัติได้ถูกบันทึกไว้ใน CD ที่แนบมากับรายงานฉบับนี้ด้วย

5. ได้ตีพิมพ์ผลการวิจัย

ได้ผลงานทางวิชาการตีพิมพ์ลงในวารสารระดับนานาชาติ

1. Sukon Puntunan and Manukid Parnichkun. "An Online Self-Tuning Precompensation for a PID Controller: An Application to Control Heading Direction of a Flying Robot," International Journal of Advanced Robotics. Robotics Society of Japan. (ได้ถูกแก้ไขตามข้อเสนอแนะของผู้ประเมินบทความและถูกส่งไปพิจารณาเป็นครั้งที่ 2 ขณะนี้กำลังรอฟังผลการพิจารณาบทความ)
2. Vatee Pariyapong and Manukid Parnichkun. "Evolutional Sensor Fusion for an Autonomous Flying Robot," Journal of Intelligent &Fuzzy Systems. IOSPress. (ถูกส่งไปพิจารณาเป็นครั้งแรก ขณะนี้กำลังรอฟังผลการพิจารณาบทความ)
3. Sukon Puntunan and Manukid Parnichkun. "Development and Control of 6-DOF Fully Autonomous Flying Robot by Hybrid Adaptive Neuro-Fuzzy Model Reference Control," International Journal of Mechatronics. Elsevier Science Ltd., Pergamon. (กำลังแก้ไขภาษาและจะส่งไปพิจารณาเป็นครั้งแรกในไม่ช้า)

ได้ผลงานทางวิชาการตีพิมพ์ลงในวารสารในประเทศ

1. Sukon Puntunan, Manukid Parnichkun. "Control of Attitude and Heading of an Autonomous Flying Robot," Journal of the Thai Robotics Society, Vol 3, No. 1, 2005, pp. 53-65.
2. Sukon Puntunan, Manukid Parnichkun. "Control of a Floating Height and Heading Direction of an Autonomous Flying Robot," Journal of the Thai Robotics Society, Vol. 2, No. 1, 2004, pp. 1-9.

ได้ผลงานทางวิชาการตีพิมพ์ลงในเอกสารการประชุม

1. Sukon Puntunan, Manukid Parnichkun, "Self-Tuning Preecompensation of PID based Heading Control of a Flying Robot." Proceedings of IEEE Workshop on Advanced Robotics and Its Social Impacts, ARSO '05, Nagoya, 2005. (Conference CD-ROM).
2. Sukon Puntunan, Manukid Parnichkun, "An Online Self-Tuning Precompensation for a PID Controller." Proceedings of the Fourth Asian Conference on Industrial Automation and Robotics, ACIAR 2005, Bangkok, 2005. (Conference CD-ROM).
3. Sukon Puntunan, Manukid Parnichkun, "Self-Tuning Preecompensation of PID based Heading Control of a Flying Robot." Proceedings of the Fifth AIT-KIST International Joint Symposium, Seoul, Korea, 2005
4. Sukon Puntunan, Manukid Parnichkun, "Attitude and Heading Control of an Autonomous Flying Robot." Proceedings of the 2004 IEEE International Conference on Industrial Electronics, IECON 2004, Busan, Korea, 2004. (Conference CD-ROM).
5. Sukon Puntunan, Manukid Parnichkun, "Control of Attitude and Heading of an Autonomous Flying Robot." Proceedings of the 2004 TRS Conference on Robotics and Industrial Technology, CRIT 2004, Nakorn Pathom, Thailand, 2004, pp. 45-49.

6. Vatee Pariyapong, Manukid Parnichkun, “Pose Estimation of an Autonomous Flying Robot Using Evolutional Ensemble Structure of Multiple Local Sensor Fusion Networks.” Proceedings of the Third Asian Conference on Industrial Automation and Robotics, ACIAR 2003, Bangkok, 2003, pp. 116-123.
7. Sukon Puntunan, Manukid Parnichkun, “Control of Heading Direction and Floating of a Flying Robot.” Proceedings of the 1st Aeronautic and Aerospace Engineering Network Conference, Bangkok, 2003, pp. 88-91. (in Thai)
8. Vatee Pariyapong, Manukid Parnichkun, “Ensemble Structure of Multiple Local Sensor Fusion Machine Using Evolutional Pruning Technique [An Application to Heading and Rate of Turn Estimation].” Proceedings of the 2002 IEEE International Conference on Industrial Technology, ICIT’ 02, Bangkok, 2002, pp. 421-426.
9. Sukon Puntunan, Manukid Parnichkun, “Control of Heading Direction and Floating Height of a Flying Robot.” Proceedings of the 2002 IEEE International Conference on Industrial Technology, ICIT’ 02, Bangkok, 2002, pp. 690-693.

ทั้งนี้ได้มีการกล่าวในกิติกรรมประกาศถึงการสนับสนุนด้านเงินวิจัยจากสำนักงานกองทุนสนับสนุน การวิจัยของทุกผลงานที่ถูกตีพิมพ์

ภาคผนวก

ต้นฉบับบทความที่ส่งไปพิจารณาเพื่อตีพิมพ์ในวารสารวิชาการระดับนานาชาติ

1. Sukon Puntunan and Manukid Parnichkun. “An Online Self-Tuning Precompensation for a PID Controller: An Application to Control Heading Direction of a Flying Robot,” International Journal of Advanced Robotics. Robotics Society of Japan.
2. Vatee Pariyapong and Manukid Parnichkun. “Evolutional Sensor Fusion for an Autonomous Flying Robot,” Journal of Intelligent & Fuzzy Systems. IOSPress.
3. Sukon Puntunan and Manukid Parnichkun. “Development and Control of 6-DOF Fully Autonomous Flying Robot by Hybrid Adaptive Neuro-Fuzzy Model Reference Control,” International Journal of Mechatronics. Elsevier Science Ltd., Pergamon.

AN ONLINE SELF-TUNING PRECOMPENSATION FOR A PID CONTROLLER: AN APPLICATION TO CONTROL HEADING DIRECTION OF A FLYING ROBOT

SUKON PUNTUNAN and MANUKID PARNICHKUN

Asian Institute of Technology, P.O. Box 4, Klong Luang, Pathumthani, 12120, Thailand

Tel: +66-2-524-5229, Fax: +66-2-524-5697, Email: manukid@ait.ac.th

Abstract

In this paper, an online self-tuning precompensation for a Proportional-Integral-Derivative (PID) controller is proposed to control heading direction of a flying robot. The flying robot is a highly nonlinear plant, it is a modified X-Cell 60 radio-controlled helicopter. Heading direction is controlled to evaluate efficiency of the proposed precompensation algorithm. The heading control is based on the conventional PID control combined with an online self-tuning precompensation so that both the desired transient and steady state responses can be achieved. The precompensation is applied to compensate unsatisfied performances of the conventional PID controller by adjusting reference command of the conventional PID controller. The precompensator is based on Takagi-Sugeno's type fuzzy model, which learns to tune itself online. The main contribution of the proposed controller is to enhance the controlled performance of the conventional PID controller by adding a self-tuning precompensator on the existing conventional PID controller. The results show that the conventional PID controller with an online self-tuning precompensation has a superior performance than the conventional PID controller. In addition, the online self-tuning precompensation algorithm is implemented simply by adding the precompensator to the existing conventional PID controller and letting the self-tuning mechanism tune itself online.

Keywords: Flying Robot, PID control, Fuzzy Logic, Online Self-Tuning

1. INTRODUCTION

A flying robot developed at AIT is modified from X-Cell 60 radio-controlled helicopter. It is developed to support autonomous flight control covering wide-mode missions of operation from hovering to other maneuvers. Currently, there are many researches on development of autonomous flying robots with different control techniques [5]. The conventional PID controller is still widely used due to its simple implementation and tuning. The weakness of the conventional PID controller is that it exhibits poor performance when applied to control the system that contains nonlinear and cross coupling effects. Various techniques are applied to accomplish this purpose, ranging from adjusting the controller gains to using the precompensation technique. The latter has many advantages since it is simple to implement and safe. In the precompensation technique, the controller gains are the same as the ones obtained in stable response and the precompensated amounts can be bounded within reasonable ranges of safety.

Since the introduction of fuzzy set by Zadeh [4], fuzzy logic-based controllers have received considerable interest from many researchers. Kim et al. [2] applied a fuzzy precompensated PID controller to control position of a DC servomotor by compensation of overshoots and undershoots of transient response under load variation. In our work, the conventional PID controller with an online self-tuning precompensation is used to control heading direction of our flying robot. The precompensator is based on the Takagi-Sugino's type fuzzy model. There are three main reasons to apply the precompensator to overcome the unsatisfied controlled performance. Firstly, to eliminate steady state error. Even when the integral term is included in the controller, steady state error still occurs in the results, due to many factors such as, deadzones in the linkage mechanism, slow speed and delay of the actuator, varying of rotation speed of the tail rotor and unsymmetrical yaw dynamics in clockwise and counter clockwise rotations. Secondly, to reduce cross coupling effected from the Z-axis. Lastly, to decrease settling time in the yaw dynamics response. By using online self-tuning precompensation with the conventional PID controller, the system exhibits superior transient as well as steady state performances.

The precompensation technique described in this paper is different from the precompensation addressed by Kim, et al in [2]. Firstly, the system and fuzzy model are totally different. In Kim's work, the fuzzy model is based on Mamdani's model. In this work, the fuzzy model is based on Takagi-Sugeno's model. Secondly, technique of tuning of the fuzzy logic is different. The technique used by Kim is based on manual tuning. In this work, it is based on online self-tuning by gradient descent method. Thirdly, compensation design by Kim is based on an attempt to compensate overshoots and undershoots in the transient response when the conventional PID controller is applied to a DC servo position controlled testbed with load varying. In this work, the design is based on compensation of steady state error and reduction of cross coupling effects as well as improvement of settling time when the conventional PID controller is applied to control heading direction of the flying robot.

This paper is organized as follows. In section 2, we describe architecture of our flying robot. Section 3 describes control structure of the precompensation. Section 4 describes experimental results, which demonstrate performances of the algorithm. Section 5 shows the fully autonomous flight experiment. Finally, the conclusion is made in section 6.

2. FLYING ROBOT AND FLIGHT CONTROL SYSTEM

Our flying robot is a modified X-Cell 60 radio-controlled helicopter with a main rotor diameter of 1.80 meters. The robot's OS91 glow plug engine has power rating of 3.0 HP, resulting in the maximum payload of 5.0 kg and flight duration of approximately 15 minutes. Fig. 1 shows the flying robot and its avionics box of the robot. The avionics box, which is installed underneath the robot, contains the following processors and sensors.

- An onboard PCM3350 PC-104 flight control computer running at 300 MHz.
- Two 68HC11 microprocessors. The first microprocessor generates pulse width modulation (PWM) signals to drive 4 actuators. The second microprocessor is used to drive and read an ultrasonic altimeter.

- A 3DM-GX1 attitude and heading reference sensor containing three angular rate gyros, three orthogonal linear accelerometers, and three orthogonal magnetometers to provide three orientation angles (roll, pitch, yaw).
- An OEM4 RT-20 GPS card. The GPS provides latitudes and longitudes information within 20 cm CEP (circular error probable) when operated in a real time kinematics mode.
- An SRF-04 ultrasonic altimeter to provide ground-to-robot distance at the update rate of 25 Hz.
- A circuit board containing actuator-interfacing circuit and control signal multiplexing circuit.

Fig. 1 Flying robot testbed

In the control inner loop, the PC-104 computer receives attitude information (roll, pitch, and yaw) from the attitude and heading reference sensor and runs the core PID attitude control at the rate of 50 Hz, effecting the aileron, elevator and rudder actuators. The control outer loop, the position control, is run at 5 Hz to generate the roll, pitch and yaw (heading) attitude commands for the inner loop while the height control is run at 25 Hz to control the position in Z-axis. The flying robot continuously communicates with ground station via an 802.11b wireless network using TCP/IP protocol. The communication occurs every 2 seconds and the range of communication covers up to 0.5 km. The ground station sends DGPS correction signal and updates user commands to the flying robot. Fig. 2 shows the flight control system of our flying robot.

Tail rotor on the flying robot is used to control the robot heading direction by altering pitch angle on the tail rotor blades. By doing so, it can increase or decrease the yaw angular moment of the robot. The robot actuators are the S9206 dc servomotor, accompanied with GY401 rate gyros.

To evaluate control performance of online self-tuning precompensation on the conventional PID controller. The engine governor is always turned off during the experiments.

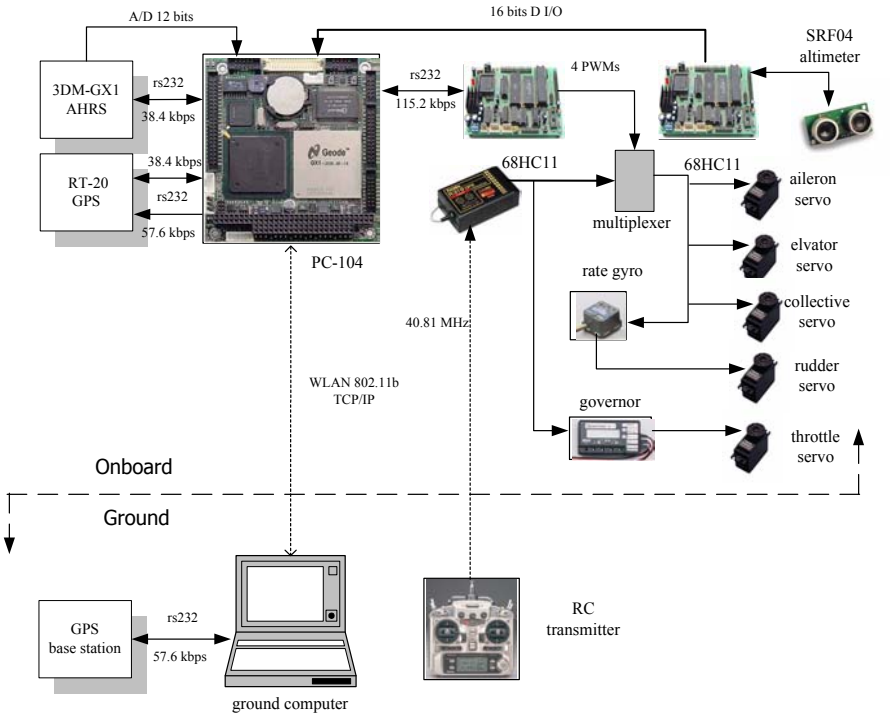


Fig. 2 Flight control system

3. ONLINE SELF-TUNING PRECOMPENSATION

Normally, control performance of a system can be improved by tuning of the controller gains. This method can harm the flying robot, since the robot flies in the turbulent air where the system parameters change all the time. The proposed algorithm in this paper applies the method that adjusts the heading reference command of the controller instead of directly adjusts the controller gains. The precompensator uses the gradient descent method to tune the fuzzy parameters. The originality on this proposed method is the use of the conventional PID controller together with the online self-tuning precompensator to control heading direction of the flying robot. The steady state error is eliminated online during the flight. The cross coupling effect to the control axis is also reduced. It is a kind of an adaptive control, since when the robot dynamics changes, the control system will tune itself and adapt to the new flight condition. The main advantage of online tuning is that it makes the development simpler in practical. Unlike in the work of Kim et al [2], where the fuzzy parameters are tuned by the operator experience to obtain the best result. In this proposed method, the fuzzy parameters are adapted based on the control performance. The process is done online automatically.

An online self-tuning precompensation for the conventional PID controller is proposed and applied to control heading direction of the flying robot. Fig.3 illustrates block diagram of the controller. The diagram consists of the conventional PID controller and the online self-tuning precompensator.

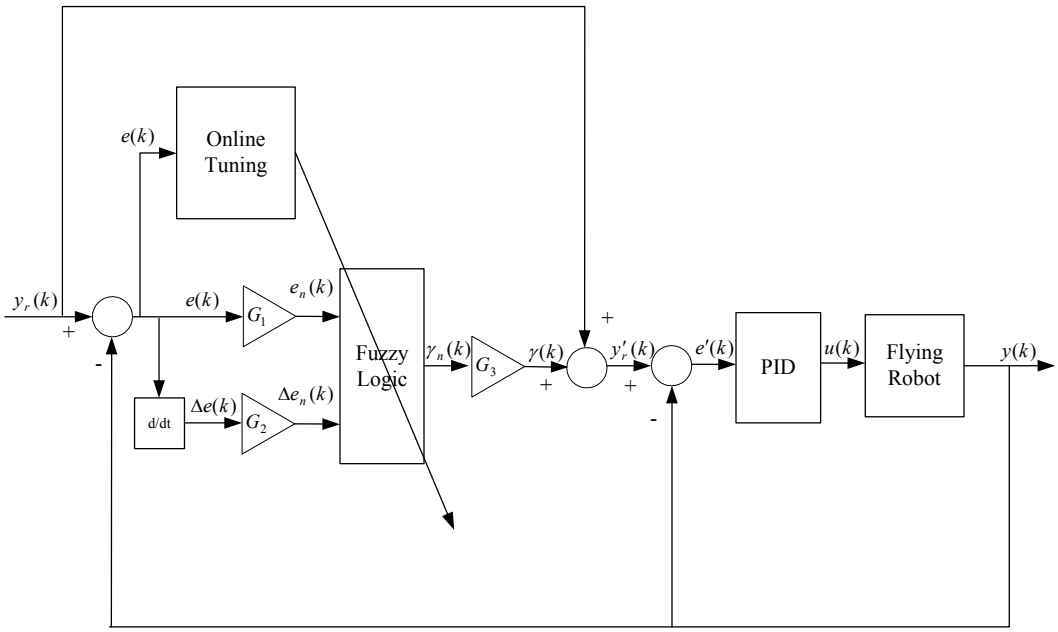


Fig. 3 Online self-tuning precompensation of PID controller

Purpose of the precompensation is to modify reference command to compensate steady state error, overshoots, and cross-coupling effects. The precompensator consists of two parts; fuzzy logic and online self-tuning mechanism. Heading error, $e(k)$, and change of heading error, $\Delta e(k)$, are determined as followings.

$$e(k) = y_r(k) - y(k) \tag{1}$$

$$\Delta e(k) = e(k) - e(k-1) \tag{2}$$

where $y_r(k)$ is the command reference and $y(k)$ is the actual output response.

Two input variables of the fuzzy logic are the normalized heading error, $e_n(k)$, and the normalized change of heading error, $\Delta e_n(k)$. They are obtained by multiplying the heading error, and the change of heading error, with their corresponding scaling factors G_1 and G_2, as followings.

$$e_n(k) = G_1 e(k) \tag{3}$$

$$\Delta e_n(k) = G_2 \Delta e_n(k) \tag{4}$$

The normalized correction value, $\gamma_n(k)$, is the result of mapping from $e_n(k)$ and $\Delta e_n(k)$ to $\gamma_n(k)$ based on Takagi-Sugeno's fuzzy model as shown in equation (5).

$$\gamma_n(k) = F\left[e_n(k), \Delta e_n(k)\right] \tag{5}$$

To obtain the actual correction value, $\gamma(k)$, the normalized correction value must be multiplied with a coefficient G_3 as shown in equation (6).

$$\gamma(k) = G_3 \gamma_n(k) \tag{6}$$

The precompensated reference command, $y'_r(k)$, is the sum of the reference, $y_r(k)$, and the correction term, $\gamma(k)$, as shown in equation (7).

$$y'_r(k) = y_r(k) + \gamma(k) \tag{7}$$

The precompensated reference command is, finally, used as the input to the conventional PID controller as followings.

$$e'(k) = y'_r(k) - y(k) \tag{8}$$

$$u(k) = u(k-1) + K_P\left[e'(k) - e'(k-1)\right] + K_i e'(k) + K_D\left[e'(k) - 2e'(k-1) + e'(k-2)\right] \tag{9}$$

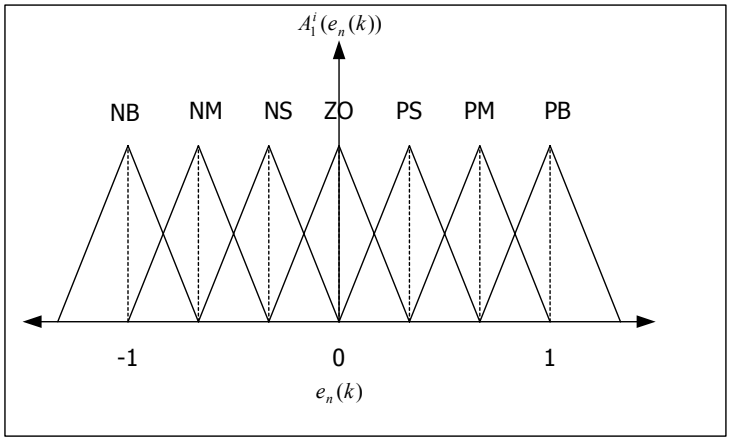
where K_P, K_I, and K_D are the proportional, integral and derivative gains, respectively.

In equation (8), the error $e'(k)$ is the tracking error between the precompensated heading reference, $y'_r(k)$, and the actual heading, $y(k)$. Equation (9) represents velocity version of PID controller. The controller output, $u(k)$, is then converted to PWM signal to drive rudder actuator of the flying robot.

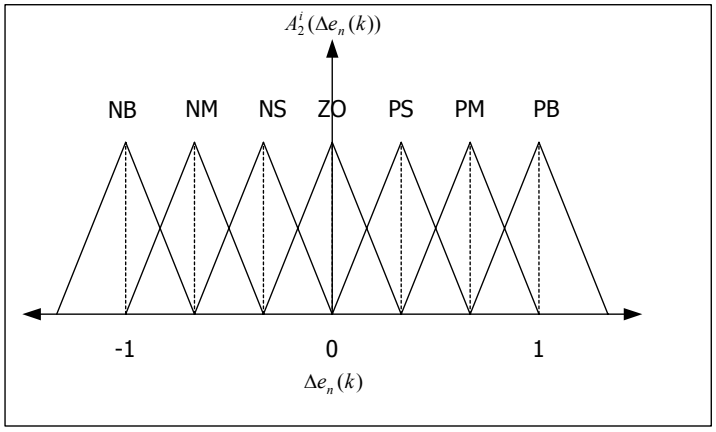
The precompensation applies Takagi-Sugino's fuzzy model. The model is formulated following the form.

$$L^i: \quad \textit{If } e'(k) \textit{ is } A_1^i \textit{ and } \Delta e'(k) \textit{ is } A_2^i \textit{ then } \gamma_n^i(k) = a_0^i \tag{10}$$

where $L^i(i = 1,2,...l)$ denotes the i-th implication, i is the number of fuzzy implication, $a_0^i(i = 1,2,...,l)$ is the consequent parameter, A_1^i and A_2^i are fuzzy sets of input membership functions. Membership functions of the inputs are shown in Fig.4. The membership functions are symmetrical triangular shape. Each linguistic value is expressed by its mnemonic; for example, *NB* stands for "negative big", *NM* stands for "negative medium", *NS* stands for "negative small", *ZO* stands for "zero", and likewise for the positive (*P*) mnemonic.



(a)



(b)

Fig. 4 Membership functions of the normalized heading error in (a)
and the normalized change of heading error in (b)

The output of fuzzy logic is calculated by the weight average method, given inputs $(e_n(k), \Delta e_n(k))$, the final output is the weight average of γ_n^i as shown in equation (11).

$$\gamma_n(k) = \frac{\sum_{i=1}^{l} w^i(k)\gamma_n^i(k)}{\sum_{i=1}^{l} w^i(k)} \tag{11}$$

where $w^i > 0$, γ_n^i is the consequent of the i-th implication, and the weight, w^i, implies the overall truth value of premise of the i-th implication calculated in equation (12).

$$w^i(k) = A_1^i(e_n(k)) \bullet A_2^i(\Delta e_n(k)) \tag{12}$$

where $A_1^i(e_n(k))$ and $A_2^i(\Delta e_n(k))$ are truth-values of heading error and change of heading error of the i-th fuzzy rule calculated in equation (13).

$$A_j^i(x_j) = 1 - \frac{2\left|x_j - a_j^i\right|}{b_j^i} \qquad i = 1,2,3,...,l;\ j = 1,2 \tag{13}$$

where x_j is the input value, a_j^i is the center of triangle, b_j^i is the width of triangle membership function as define in Fig. 5.

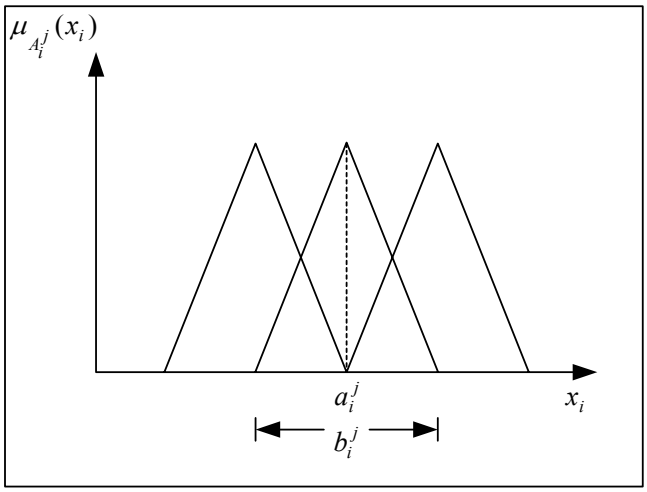


Fig. 5 Triangle membership function

The second part of the precompensator is the online self-tuning mechanism. The self-tuning method of the precompensator applies gradient descent technique. The precompensator is tuned by minimizing a cost function. The cost function is defined as the square of the difference between the actual heading and the reference command as expressed in equation (14).

$$E = \frac{1}{2}(y - y_r)^2 \tag{14}$$

In the self-tuning, only the parameters in the consequent part of the rules are updated. Self-tuning of the precompensator parameters, a_0^i, by gradient descent method is expressed by equation (15).

$$a_o^i(k+1) = a_0^i(k) - \eta \frac{\partial E}{\partial a_0^i} \tag{15}$$

where η, is a non-negative learning rate, a_0^i is tuned by equation (15). The gradient of the cost function with respect to a_0^i parameter is calculated from equation (16).

$$\frac{\partial E}{\partial a_0^i} = \frac{w^i(k)}{\sum_{i=1}^{l} w^i(k)}(y_r(k) - y(k)) \tag{16}$$

4. FLIGHT EXPERIMENT RESULTS

In the experiments, performances of the conventional PID controller and the PID controller with online self-tuning precompensation are compared. The PID gains, which result in a satisfactory system performance, are the result of trial and error of many experiments. Finally, the PID gains used in the experiments are K_P=10, K_I=0.0125 and K_D =6.4. The fuzzy logic consists of 49 rules. The heading reference is compensated when the heading error is in the range of ±80 degrees. So, scaling factor for the

heading error is selected at $G_1 = \frac{1}{80}$ to ensure that the overall normalized heading error is in the applicable boundary of the fuzzy input. The change of heading error range is limited within ±5 degrees. So, the scaling factor for the change of heading error is selected at $G_2 = \frac{1}{5}$ to ensure that overall normalized change of heading error is in the applicable boundary of the fuzzy input. For safety reason, the correction output is bounded within ±10 degrees by applying the output scaling factor at $G_3 = 10$. For simplicity, the centers of the input membership functions of NB, NM, NS, ZO, PS, PM and PB are selected at the points–1.0, -0.66, -0.33, 0, 0.33, 0.66, and 1.0, respectively. The learning rate is selected at 0.5. The same PID gains are used in both the conventional PID and the PID controller with the online self-tuning precompensation. The initial values of all the consequent parts of the fuzzy logic are set at zero. It means the correction of zero at the beginning.

$$a_0^i = 0 \qquad \text{for all } i \tag{17}$$

The heading control loop and the precompensation loop in the experiment are run at 50 Hz. Fig. 6 shows results of the flight experiments. The centers of the output membership functions at the end of the self-tuning process are shown in Table 1. Fig. 7 shows the fuzzy output of the precompensator. In Fig. 7(a), the correction value is zero because all of the consequent parts of the fuzzy logic are initialized at zero. Fig. 7(b) shows the outputs of the precompensator after the tuning process is done.

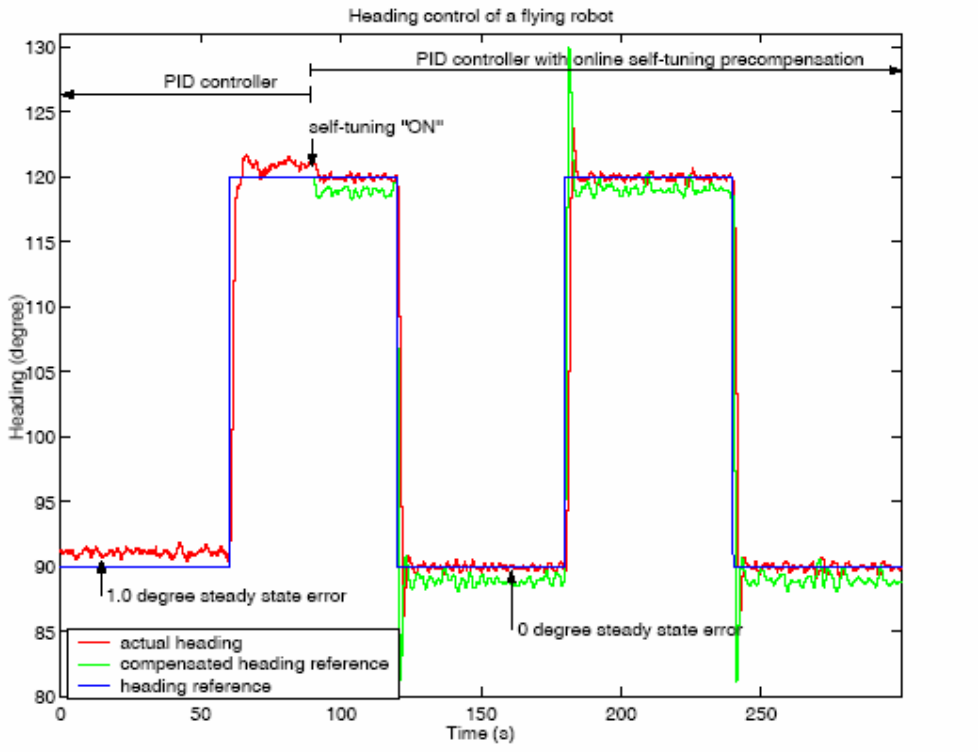


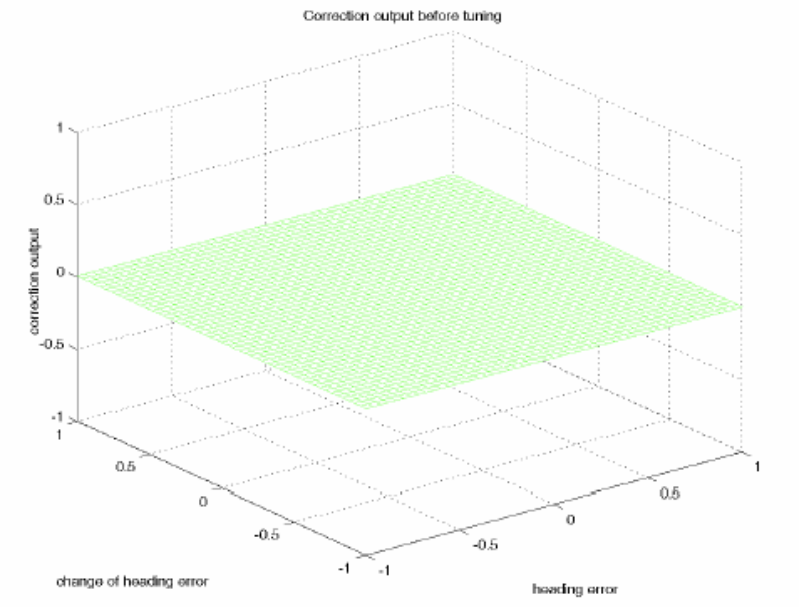
Fig. 6 Heading control experiment

Fig.6 illustrates significantly improvement when the online self-tuning precompensation is turned on at time t = 90 seconds. At the beginning, only the conventional PID controller is applied, it results in a steady state error in the output response. By the precompensation, the steady state error is eliminated.

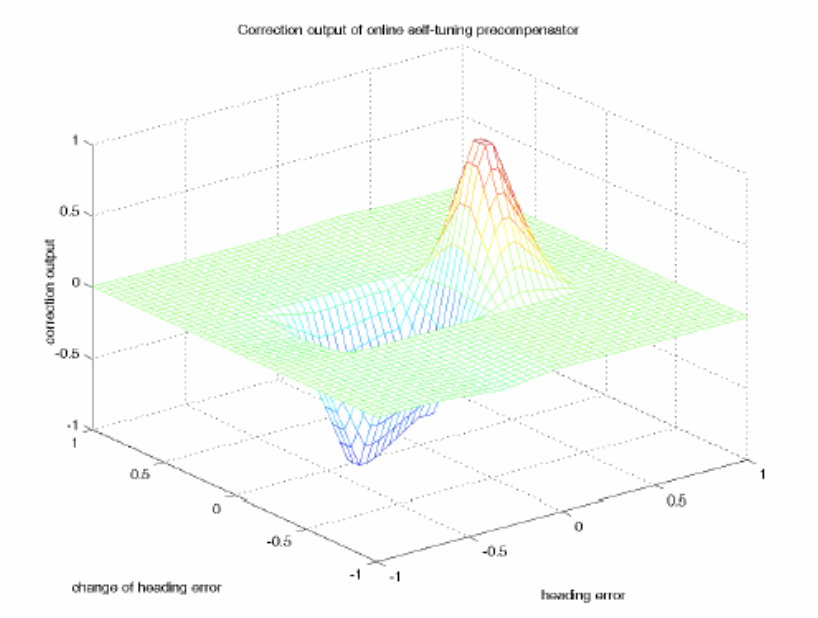
Table 1 Consequent parameters of fuzzy logic after tuning

$\Delta e_n(k)$ \ $e_n(k)$	NB	NM	NS	ZO	PS	PM	PB
NB	0	-0.0034	-0.0426	0	0	0	0
NM	0	-0.0022	-0.0279	0	0	0	0
NS	0	0	0	0	0	0	0
ZO	0	-0.0368	-1.0	-0.7795	1.0	0.0198	0
PS	0	0	-0.0046	-0.0007	0	0	0
PM	0	0	0	0	0.0140	0.0010	0
PB	0	0	0	0	0.0214	0.0015	0

Fig. 8 (a) shows the effects of cross coupling from Z-axis of flying robot to the heading control performance. Firstly, the precompensation algorithm is turned off. The flying robot takes off and head to 120 degree by the conventional PID control. The flying robot then rapidly changes its altitude from 1 meter to 3 meters and changes back to 1 meter again. By the conventional PID controller, the heading moves away from the setpoint to 50 degree in the counter clockwise direction, which is 70 degrees away from the setpoint. The similar experiment is conducted on the flying robot again by using the online self-tuning precompensation. The result is shown in Fig. 8 (b). The precompensation significantly reduces the effect of cross coupling.



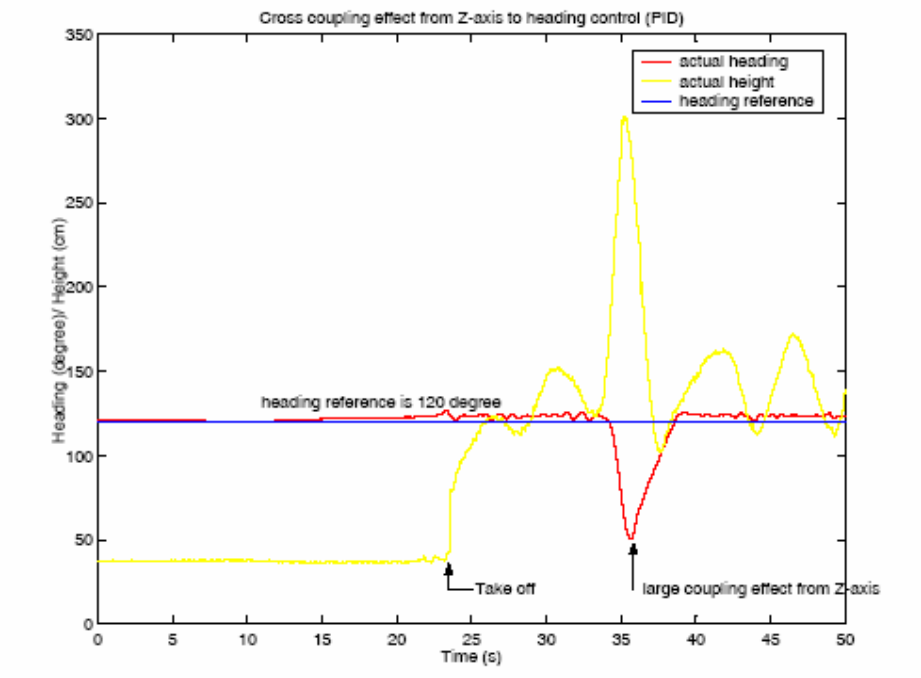
(a)



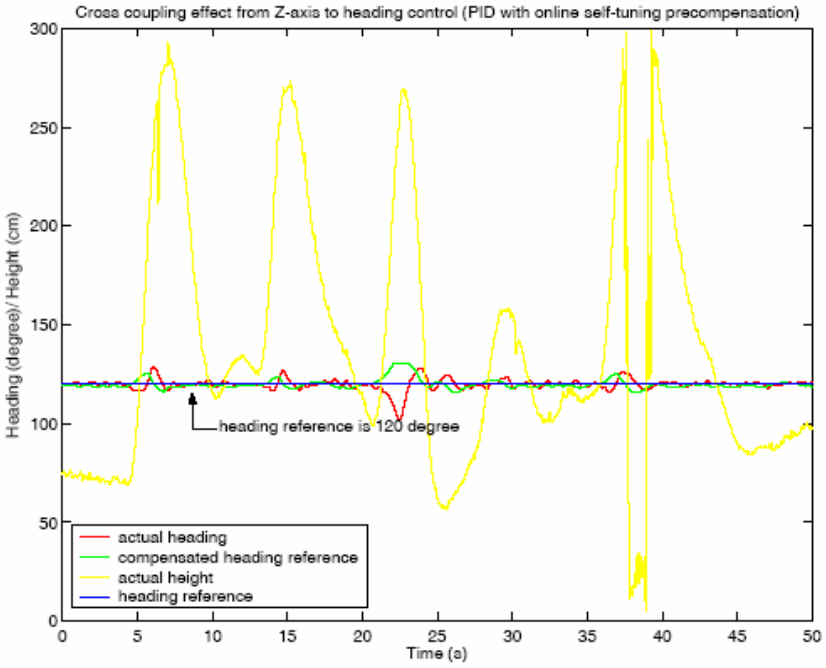
(b)

Fig. 7 Compensation output (a) before tuning and (b) after tuning

The other experiment tests the effect of self-tuning as shown in Fig. 9. The flying robot lands at time t = 5 seconds. During landing, self-tuning is saturated. Then the flying robot takes off again at time t = 32 seconds. The result shows the efficiency of the online self-tuning mechanism of the precompensation.



(a)



(b)

Fig. 8 Effects of cross coupling from Z-axis of flying robot to heading control by (a) PID alone and (b) PID with online self-tuning precompensation

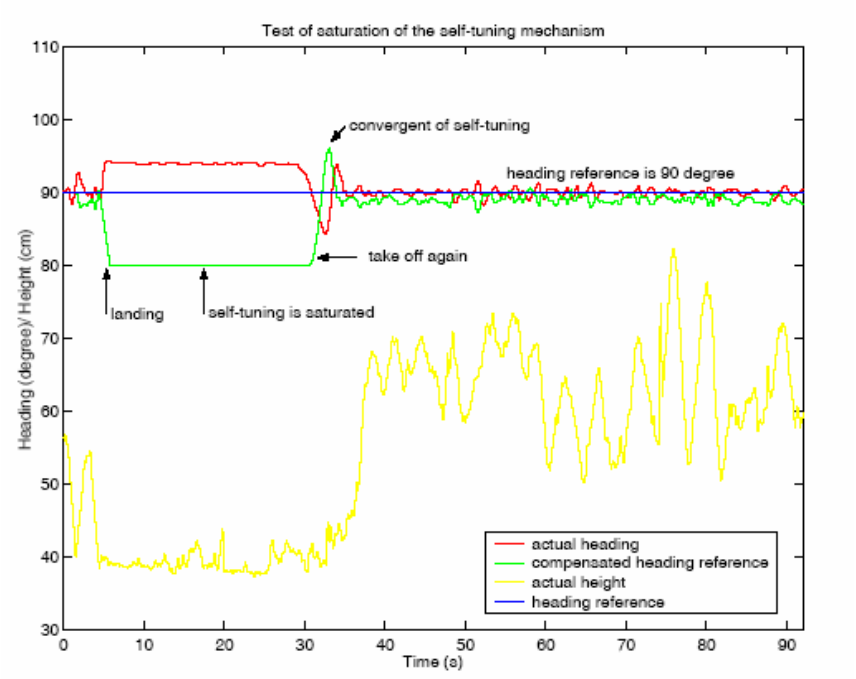


Fig. 9 Saturation of an online self-tuning precompensation

5. AUTONOMOUS FLIGHT TEST

In order to test the control performance of the proposed algorithm, the flying robot is commanded to fly in a 3 meters by 3 meters square area. The flight experiment is shown in Fig.10. There are 4 points marked with the white spot on the ground. The flying robot is automatically controlled in 6 DOF, including roll, pitch, yaw, X-axis position, Y-axis position and Z-axis position. The altitude command is 1.5 meters above the ground. The X-Y position commands are changed sequentially among the 4 marked points. The X-Y position commands are changed every 30 seconds. The heading command is maintained at 0 degree (pointing to the North). Fig.10 shows the flying robot tracking to the marked points. In the fully autonomous flight, the heading control was disturbed by air turbulent and ground effect. From the results, the zero degree heading command can be maintained with good performance. Fig. 10 (a) shows the robot flying over the first point. Fig. 10(b) shows the robot flying to the second point. Fig. 10 (c) shows the robot flying over the second point. Fig. 10(d) shows the robot flying to the third point. Fig. 10(e) shows the robot flying over the third point. Fig. 10(f) shows the robot flying to the fourth point. Fig. 10(g) shows the robot flying over the fourth point. Lastly, Fig. 10(h) shows the robot flying to the home point and the mission is accomplished.

(a)

(b)

(c)

(d)

Fig. 10 Autonomous flight test with heading point to 0 degree

6. CONCLUSION

In this paper, the conventional PID controller with and without online self-tuning precompensation were applied to control heading direction of a flying robot. The conventional PID controller was used as the basis control method. The adaptable component that used as the command precompensator was successfully applied. There are two advantages of the proposed method in improving the control performance of the conventional PID controller. Firstly, the existing PID controller is still used. Secondly, the precompensator parameters are tuned automatically online. The precompensation is very simple to be integrated into the existing conventional PID controller. This is simply done by adding the precompensation in front of the existing PID controller. The performance was evaluated by many flight experiments. The results demonstrated good performance of online self-tuning precompensation. The conventional PID controller with online self-tuning precompensation provided much better responses compared to the conventional PID controller alone. The steady state error was eliminated. The effect of cross coupling was reduced. The settling time was decreased. However, as shown in the results, there existed overshoots in the output responses. This was because of the fixed learning rate. In order to achieve a better result, variation of the learning rate is required.

7. ACKNOWLEDGEMENT

This research project is financially supported by Thailand Research Fund.

REFERENCES

[1] G. Padfield. Helicopter flight dynamics. Blachwell Science Ltd, 1996.

[2] J. -H. Kim, K. -C. Kim, and S. -W. Lee. Fuzzy precompensation of PID controllers. The IEEE conference on control applications, 1993.

[3] J. -H. Kim, J. -H. Park, S. -W. Lee, and E. K. P. Chong. Fuzzy precompensation of PD controllers for systems with Deadzones. Journal of intelligent and fuzzy system, 1993.

[4] L. A. Zadeh. Fuzzy set. Informat. Control; Vol. 8, 1965.

[5] S. Saripalli, J. M. Roberts, P. I. Corke, G. Buskey. A tale of two helicopters. Robotics research lab, University of Southern California, 2002.

Evolutional Sensor Fusion for an Autonomous Flying Robot

VATEE PARIYAPONG[1], MANUKID PARNICHAKUN[2]

[1] Phd. Student, Mechatronics, School of Advanced Technologies, Asian Institute of Technology, Thailand. E-mail: rtafa40@hotmail.com, vatee@genex-school.com, Tel: 669-133-7709, Fax: 662-524-5697.

[2] Associate Professor, Mechatronics, School of Advanced Technologies, Asian Institute of Technology, Thailand. E-mail: manukid@ait.ac.th, Tel: 662-524-5229, Fax: 662-524-5697.

Abstract

In this paper, we propose an approach based on a mixture of divide-and-conquer principle, Radial Basis Function network (RBFN), and Evolutional Algorithm (EA) to fuse highly corrupted data from a digital compass with those from a rate gyro and an angular accelerometer such that the reliability and robustness on the heading information of an autonomous flying robot is greatly improved in the sense that the uncertainty hyper-ellipsoid of fused data is minimized. The whole fusion process is taken as a generic, time-invariant, nonlinear dynamics model which relates a sensory (raw) data input vector to a fused output as the heading position of such a robot. The architecture contains two hierarchical levels: local and global fuser engine; hence decomposing design process into two independent steps. Two key algorithms called "Hybrid GA/OLS (HGAOLS)" and "Evolutional Ensemble Averaging (EEA)" are composed for a purpose of local and global fuser construction process. The resulting fusion network shows a great deal of improvement when compared with an original digital compass in aspect of robustness to statistical uncertainty of the sensors, modular network structure, and fault-tolerance. Also our fusion approach outperforms conventional fusion methods (e.g. kalman filter and Bayesian filter) in that it is model-free and adaptive to a changing environmental condition. The approach is considered as a generic model which can be implemented to any kind of sensor platform. Also, unlike conventional methods, an assumption on a known statistics of the sensors is not necessary since it can be learned iteratively. So a set of raw data can be directly fed through the fusion engine and the fused data is obtained without any prior sensor error distribution knowledge.

Keywords – *Sensor Fusion, Divide-and-Conquer, Evolutional algorithm, Radial Basis Function network*

1. INTRODUCTION

A digital compass is commonly used to determine heading information of an autonomous robot. However, in practice compass itself possesses some limitation. Its accuracy often depends on several conditions; such as the robot movement, magnetic field around the sensor module, sampling speed, and sometimes the vibration of the robot structure itself. Additionally, the nonlinearity during a conversion process from raw sensory data to heading information usually occurs. To cope with such an undesirable effect, a sensor fusion process is implemented by integrating other kinds of sensors with the compass such that fused information is more accurate. Majumder [15], for example, fuses a set of internal sensors (rate gyro, accelerometer,) with external sensors (digital compass, and pressure sensor) to extract a feature of unstructured environment around an autonomous robot. In our work, the heading data from digital compass is fused with the velocity data from a rate gyro and the acceleration data from an accelerometer to better estimate the heading information of a flying robot (shown in Fig. (1)).

Sensory Data Fusion (SDF) has become a promising technology for the robotics community. For years, both of the statistical and probabilistic fusion methods have been concurrently developed by many researchers [9], [21]. In the statistical approach, a fused feature of the target object is obtained in a framework of statistical detection theory. By doing so, with a set of noisy measurement from different sensors, a fused data is found such that it minimizes the integral of probability of unacceptable error. Among all, kalman filter and its variants ([20], [23], and [12]) are considered the most widely accepted techniques due to their computational simplicity and hardware memory saving. However, the main drawback of kalman fileter-based fusion is that it is model-based technique; i.e., the state and measurement equation must be provided *a priori*. Furthers, the complete knowledge of noise distribution both on the process and sensor measurement is a must in order to implement the kalman estimator effectively. The whole problem gets worse, when the state and/or measurement equation is intrinsically nonlinear. An optimal filter results can no longer be guaranteed. While in probabilistic approach, sensor fusion has its root from Bayesian rule of inference. The fused data is the one that maximized the posterior output distribution conditioned on a given set of current sensor measurements. Normally, in literatures, this leads to the ML (Maximum Likelihood) and MAP (Maximum A Posteriori) estimator [1]-[2]. In spite of the usefulness of ML and MAP, like the statistical approach, Lua and Su [14] argued that the pitfall of the probabilistic fusion is the fact that the requirement of complete (or partial) statistical information of the sensor is inevitable during the filter computation. Unfortunately, no such sensor can, in fact, be completely represented by its statistical characteristics in the real world due to a non-uniform error distribution of such a sensor. Besides, most of the practical sensors need calibration process before being put in use, and it is often not an easy task to do so. Consequently, this makes the sensor modeling problem more involved than the sensor fusion problem itself. Sensor modeling problem gets even harder when the sensor statistics is highly sensitive to the environment around it.

In recent years, intelligent learning framework of sensor fusion has been applied as a counterpart to the statistical and probabilistic approach [8], [10]-[11], and [24]-[25]. One of the merits of such techniques over the statistical and probabilistic methods is that the statistical information (probability distribution) of each sensor is now only a sufficient condition. In principles, by considering the fusion engine as a (possibly) nonlinear parameterized function that acts as an

expected value of fused data conditioned on a given input vector from sensory measurement, $y_{fused} = E[y \mid x] = f(x)$, an ultimate goal is to train such a fusion model to learn for intrinsic property of the underlying distribution (nonlinear conversion from raw to fused data ouput, and measurement error distribution). For realizing learning mechanism, Artificial Neural Network (ANN) model is a common method due to its nonlinear learning capability. Whereas in aspects of high-level information fusion process, Fuzzy Logic (FL) and possibility theorem, like Dempster-Shafer theory (DS; [25]), are quite popular. Both FL and DS are capable of dealing with vagueness in data both qualitatively and quantitatively. However, ANN, FL, and DS possess one major drawback. Their required network size (complexity) to effectively estimate an underlying process grows exponentially larger as the dimension of input vector to the model is increased.

Following the intelligent learning framework, in this paper, a sensor fusion system based on the mixture of "*Divide-and-Conquer"* [7] principle, RBFN [3], and EA [16][26] is designed. Based on an idea of *"Divide-and-Conquer"*, instead of a (commonly complicated) single sensor fusion model (RBFN), we come up with a combination of a set of smaller, yet simpler, fusion sub-networks, each of which works best in their corresponding part of the data space. In literatures, an approach being applied is similar to ensemble or committee machines ([6], [22]) and Bayesian Model Averaging [19] concept in machine learning community. The only difference is that we apply our concept to sensor fusion problem. In doing so, we generalize a sensor fusion engine as a generic nonlinear, time-invariant, parameterized function model (i.e. RBFN) that represents a mapping from an input space (a set of raw data from physical sensors) to an output space (the fused information). All the parameters contained inside the fusion network model are adjusted iteratively to capture the intrinsic properties of the underlying system based upon a global optimization technique, i.e. GA. According to neural network learning theory, our proposed fusion network is proved to best trade-off the two heuristic problems: *Curse of Dimensionality,* and *Bias/Variance Dilemma* ([17] in an intelligent fashion. Furthermore, our final fusion network is fault-tolerant, robust, and adaptive to a changing environmental condition, as one will see on the empirical results.

The organization of this paper is as follows. In Section 2, we put our sensor fusion process into a framework of geometric sensor fusion problem. Some issues will be discussed to point out our current requirements for our sensor fusion design problem In section 3, the sensor fusion architecture is proposed and its unique characteristics are posted. The two algorithms (HGAOLS, EEA) are then derived in section 4 and 5. Next, experimental study is performed on our heading motion test-bed to verify our proposed methods in section 6. In the last section, concluding remarks are given to close our discussion and recommendation on how to extend our proposed architecture to a more general problem.

2. SENSOR FUSION PROBLEM FORMULATION

2.1 Physical sensor and logical Sensor

Consider a sensory information module, *s.i.m*, as a logical sensor that represents a nonlinear parametrized mapping , $f : R^m \rightarrow R^n$, from a raw data vector obtained from a set of physical sensors, $\boldsymbol{S} \in R^m$, to sensory information vector, $\boldsymbol{X} \in R^n$ as follows:

$$X = f(S, \Sigma) \tag{1}$$

where Σ denotes a set of parameters contained inside the *s.i.m*. In reality, the values of Σ affects the accuracy of such a *s.i.m*. In order for X to be uniquely classified, it is necessary that $n \leq m$, otherwise X cannot be determined completely. Further, S is assumed to be a measured sensory data vector which is corrupted by zero mean gaussian white noise as follows:

$$S = \bar{S} + \delta S; \quad \delta S \Leftrightarrow N(\delta S, 0, Q)$$
$$Q = diag(\sigma_i^2); \; i = 1..m \tag{2}$$

Eq. (2) assumes that all the sensory data elements are statistically uncorrelated. Based on the assumption above, we can compute the mean and error variance matrix of X as:

$$E[X] = f(\bar{S}) = \bar{X} \tag{3}$$
$$V[X] = E[(X - \bar{X})(X - \bar{X})^T] = E[J(\bar{S})\Delta S \Delta S^T J(\bar{S})^T]$$
$$= J(\bar{S})QJ(\bar{S})^T = \mathrm{H}, \qquad J = \frac{\partial f(S)}{\partial S} \; \varepsilon \, R^{nxm} \tag{4}$$

From Eq. (3), one can easily see that even though the sensor noise elements from a set of sensor measurements are statistically independent, the final information data elements are not. This is due to the nonlinear transformation given in eq.(1). If we do some further analysis by applying Singular Value Decomposition (SVD) technique on H in Eq. (4), we can say that H represents an uncertainty hyper-ellipsoid of *s.i.m*. in Eq. (1), where each singular value, σ, represents the length of each principle axis of the ellipsoid with the rotation confined by the diagonal matrix of those singular values. So if the system contains more than one *s.i.m*, the next task is how to combine this uncertain information such that the resulting fused data provides the smallest uncertainty ellipsoid as possible.

2.2 Geometric sensory information fusion

Now suppose that the information data is derived from p different *s.i.m*.. To simplify the analysis, we propose a solution on sensor fusion problem for these p sensor modules as a weighted linear combination of all individual *s.i.m*.

$$X = \sum_{i=1}^{p} W_i X_i \tag{5}$$

If we make an assumption that each *s.i.m*. is calibrated and all eventually give the same true value, $\bar{X}$, so we can assure that an expected value of the fused information, *E[X]*, will be equal to the true value. This assumption leads us to an important constraint on the weighting matrix as:

$$\sum_{i=1}^{p} W_i = I_p \tag{6}$$

where I_p is a *(pxp)* identity matrix. Using the relationship in Eq. (4) and Eq. (5), we find an error covariance matrix of the fuse information data as:

$$V[X] = E[(X - \bar{X})(X - \bar{X})^T] \tag{7.1}$$
$$= \sum_{i=1}^{p} W_i J_i Q_i J_i^T W_i^T \tag{7.2}$$
$$= \sum_{i=1}^{p} W_i \mathrm{H}_i W_i^T \tag{7.3}$$

From Eq. (7.3), our optimal weighted linear combination of these *s.i.m.* can be found by solving for an optimum weight matrix W such that Eq. (7.3) is minimized, based on the weighting matrix constraint in Eq. (6). To do so, we apply Lagrange multiplier technique on such a problem, we get an optimal weight matrix as:

$$W_{i_{opt}} = \left(\sum_{i=1}^{p} (J_i Q_i J_i^T)^{-1} \right)^{-1} \left(J_i Q_i J_i^T \right)^{-1} \tag{8.1}$$

$$= \left(\sum_{i=1}^{p} \boldsymbol{H_i}^{-1} \right)^{-1} \left(\boldsymbol{H_i} \right)^{-1} \tag{8.2}$$

Substitute W_i in Eq. (8.2) into Eq. (7.3), the error covariance matrix of **X** becomes:

$$V[X] = \left(\sum_{i=1}^{p} \boldsymbol{H_i}^{-1} \right)^{-1} \tag{9}$$

Regarding of Eq. (9), each individual H matrix is computed based on an important assumption that the statistics for each physical sensor is known *a priori*. Unfortunately, this rarely happens in the real-world application since the statistics is hard to be computed precisely (due to non-gaussian property on the uncertainty and noise distribution), and the only estimated value from an experiment is available at-hand. Additionally, the statistics could sometimes change over different operational conditions. The statistics of TCM2-50 digital compass model in our experiment, for example, changes with respect to the tilt angle and electromagnetic field condition around it. So an uncertainty in the values of H itself will lead to an erroneous result if one tries to use a fixed relation in Eq. (8.2). Further, taking a closer look for the solution of Eq. (8.2) once again, one can easily see that the existence of an optimal W lies on the critical condition that individual JQJ^T is a non-singular matrix; hence the choice of *s.i.m.* greatly affects the stability of such the solutions. Consequently, we can conclude that W_{opt} is, therefore, a complex function of a chosen *s.i.m.*, and the current condition of the statistics of a set of sensors being used, H. This can be represented mathematically as:

$$W_{opt} = G(f(S), S, H) \tag{10}$$

So an important contribution of this paper is as follows: *based on the idea of geometric sensor fusion above, we propose a sensor fusion architecture and a way to solve for W_{opt} in Eq. (10). Later, we will decompose the process into 2 steps; i.e., to first find the best f(S) for each local s.i.m. to make sure that the stability condition for the optimal solution exists, and secondly to search for a set of W_{opt} which is robust to an uncertainty for the statistics of the sensors at various conditions.*

3. SENSOR FUSION ARCHITECTURE

In this paper, our proposed sensor fusion engine contains the following key characters:

- Almost all non-ideality of physical sensors in used is captured.
- A resulting fusion topology is fault-tolerant, and robust to uncertainty for sensor statistics.

- It can be easily extended with a new sensor set or modules. In other words, its architecture will be modular as much as possible.

The architecture used for our sensor fusion engine is shown in Fig. 2 (b). We select a model known as *"Hammerstein model"* [13], providing a general nonlinear time-invariant parameterized model (Fig. 2 (a)) which maps a set of sensory (raw) data to a fused data output. Speaking of modularity property, the architecture comsists of series of nonlinear static mapping and linear filter model. For sake of ensemble networks idea, we use a weighted linear combination of multiple nonlinear mapping to fully describe nonlinearity properties in a local sense. In other words, with such architecture, each nonlinear mapping acts as a local *s.i.m.* defined in Eq. (1), while the combination serves as an ensemble averaging among these local *s.i.m.* to minimize uncertainty hyper-ellipsoid based on Eq. (10). Furthermore, the linear filter portion is implemented just to capture the dynamics of the overall sensor dynamics. Theoretically, the complexity and flexibility of the sensor fusion engine depends mainly upon the type and structure of both of the two portions. In our work, RBFN is selected as a local *s.i.m.*, and an ARX model as the linear filter (Fig. 3). The RBFN is twofold. It is used to represent all local properties (possessed by each individual physical sensor) via its receptive field width mechanism; hence a smooth transition occurs while the data point is moving across different operational regions. Also, the RBFN is proved to be a universal nonlinear function approximator with various levels of accuracy. So, as a complete structure, a discrete model of the fused data output from the fusion engine at fixed time step, k, is as follows:

$$y(k) = \sum_{i=1}^{N} a_i y(k-i) + \sum_{j=1}^{M} b_j u(k-j)$$

where

$$u(k) = \sum_{i=1}^{P} w_i f_i[S(k)] = \sum_{i=1}^{P} w_i \sum_{j=1}^{L} \alpha_j \Phi_j[S(k)] \qquad (11)$$

where *y(K)* is defined as a fused output data at the current time step, *F(S)* as a local *s.i.m.* (RBFN), *w* as weighting constant for each local fuser as in Eq. (5), *Φ(S)* as a basis function contained inside RBFN, and *S(K)* as a sensory raw data at time step, *K*, respectively. The paprameter *a*, *b*, *w*, and α are all the weighting constants. Noting that from Eq. (11), an order of the linear dynamic portion (N), the suitable number of local *s.i.m.* *(P),* and the optimal values of *w* and α are all to be determined later by our proposed algorithm.

4. LOCAL FUSION NETWORK CONSTRUCTION

The RBFN as a local *s.i.m.* provides some nice properties as mentioned previously. However, it contains one major drawback. Its complexity (network's size) grows exponentially while the dimension of input vector gets larger. In this paper, we propose HGAOLS algorithm to solve such a problem by incorporating EA approach into a design process. Also, during a desing process, we use a basis function of RFBN as a gaussian function with dead zone on the top, shown in Fig. (4). The principle behind this is to let each basis function imitate a range of physical reliability of different physical sensors; i.e., it remains constant around an nominal point and continues to drop as the distance grows from that nominal point. By this means, we will consider a set of centers of each individual basis function, radius of the dead zone

area, and the receptive field width as a set of structural parameters that need to be learned during the training process of the RFBN.

4.1. Hybrid GA/OLS: OLS with Adaptive Structure.

The OLS algorithm and its modified version were derived in [4] as a forward regression procedure to select a suitable set of centers (regressors) of each individual basis function from a large set of candidate points. Normally, those candidates are slected from a pool of training samples at each step of the regression, the increment to the explained variance of the desired output is maximized by incrementing a number of orthogonal bases to the network. The OLS is proved to be an excellent method for constructing a parsimonious RBFN. Unfortunately, an optimum solution exists only if all the structural parameters mentioned eariler are known or chosen *a priori*; hence different sets of structural parameters chosen come with different performance in the final network. To find an optimal set of structural parameters is yet another critical issue. In our approach, we employ a full advantage of the OLS with further development to solve its pitfall. In words, an optimal solution on both of the number of most significant basis functions and the basis function structure itself is solved simultaneously by our proposed HGAOLS algorithm. Referring to Fig. (4), HGAOLS works as follows: *on every iteration of the GA, all of the structural parameters of each basis function as a chromosome of the GA process are created, and each individual candidate undergoes an original OLS algorithm to obtain individual optimal network. Random recombination and mutation are applied in order to generate a pool of parental chromosomes for the next generation, the chromosomes for next generation are selected based on their fitness function (MSE on a set of testing data), and the whole process is repeated until the stopping criterion is satisfied*. Noting that we use a floating version of GA to fit with a current problem and we implement an intermediate combination on a pair of randomly chosen parental chromosomes in order to have a smooth transition between the two candidates. For sake of the mutation mechanism, a mutation with Gaussian distribution is applied on each individual process parameters.

5. GLOBAL FUSION NETWORK CONSTRUCTION

5.1. Ensemble Machines in Geometric Sensor Fusion problem

This section deals with Eq. (5); i.e., how to fuse a set of output from local $s.i.m.$ (RBFN) such that the uncertainty hyper-ellipsoid is minimized. In machine intelligent learning community, people tackle a problem of how to combine differently trained networks in various ways (e.g., committee machines, support vector machines, ensemble machines, and Bayesian Model Averaging). Regardless of the techniques, an ultimate goal of combining multiple networks is to reduce uncertainty in term of smaller error variance matrix (uncertainty hyper-ellipsoid), while maintaining an accuracy of the overall system. It is proved that ensemble of multiple neural networks always outperforms a single neural network. However, an optimal set of weighting constants must be calculated based on the training data. In this paper, our work has the same principle as in [18], except that we apply such a problem around a sensor fusion design framework. Based on weighting constraint in Eq. (6), modify Eq. (5) by adding and subtracting the true value of information, $\overline{X}$, on the right hand side as follow:

$$X = \sum_{i=1}^{p} W_i (X_i - \overline{X} + \overline{X}) = \sum_{i=1}^{p} W_i (X_i - \overline{X}) + \overline{X} \sum_{i=1}^{p} W_i$$

$$= \overline{X} + \sum_{i=1}^{p} W_i m_i \qquad (12)$$

where m is defined as a miss or an error on each individual local ***s.i.m.*** (RBFN) compared with the true value $\overline{X}$. The error variance is recalculated as

$$V[X] = E[(X - \overline{X})(X - \overline{X})^T] = \sum_{i=1}^{P} \sum_{j=1}^{P} W_i W_j E[m_i m_j^T] \qquad (13)$$

$E[m_i m_j]$ represents an error cross-covariance matrix between the i^{th} and j^{th} local ***s.i.m.*** on a particular point of information. An optimum value of W's can also be determined again by Lagrange multiplier technique as:

$$W_{i\,opt} = \left(\sum_{j=1}^{P} \sum_{k=1}^{P} E[m_j m_k^T] \right)^{-1} \sum_{j=1}^{P} E[m_i m_j^T]^{-1} \qquad (14)$$

W_{opt} in Eq. (14) is the same as that in Eq. (8.2) only with different interpretation. Eq. (8.2) posts an existence of W_{opt} *w.r.p.* to each individual local ***s.i.m.*** through its variance matrix, JQJ^T, whereas Eq. (14) puts more focus on the statistical relationship between two local ***s.i.m.***. Consequently, an additional assumption for the existence of W_{opt} is that the cross-covariance matrix between any pair of local ***s.i.m.*** must be invertable; i.e, any two local ***s.i.m.*** (RBFN) must be as statistically independent as possible in order to have mainly diagonal covariance metrix. Unfortunately, in practice, as the number of local ***s.i.m.*** (RBFN) gets larger, there are often some of the networks that are quite similar in performance, which make the cross-covariance matrix close to singular.

5.2 Evolutional Ensemble Averaging (EEA): Multi-Objective Ensemble Machines

In this section, we construct an EEA algorithm to solve for each weighting matrix W in Eq. (8.2) or in Eq. (14) such that Eq. (13) is minimized. Our EEA algorithm is an GA-based ensemble averaging technique. It helps us avoid computational burden by avoiding direct calculation on an inversion of error cross-covariance matrix and to be assure that every local ***s.i.m.*** is as dissimilar from each other as possible. Also, it allows a linear filter portion of the sensor fusion engine in Fig. (5) to be found simultaneously.

To achieve our design objective, instead of combining all the local ***s.i.m.***, the EEA algorithm allows some of local ***s.i.m.*** to be activated or de-activated at sometime, depending upon the current performance of the final fusion engine (indicated by the fitness function of GA). Noting that our fitness function must indicate that the final sensor fusion network contains the smallest size (total number of basis functions) as possible, while maintaining the performance (generalization error on unseen data) at a reasonable level. Further, the final fuser engine should, at least, outperform the

best sensor model and each individual local ***s.i.m.*** in the mean square sense. Following these concept, the EEA algorithm can be though of as a *"multi-objective"* optimization technique. It minimizes our proposed fitness function as in Eq. (15).

$$fitness = \left(\sum_{i=1}^{P} \sum_{j=1}^{P} W_i W_j E[m_i m_j^T] \right) \times \left(T \ln(n_p + 2) \right) \times \left(\frac{1}{T} \sum_{k=1}^{T} \sum_{i=1}^{P} \beta_{ik} \right) \quad (15)$$

where n_p be the total number of basis function contained in each activated RBFN, and T be the size of training data respectively. The first term on the right hand side of Eq. (15) represents an original performance measurement in Eq. (13) which, in turn, reflects the statistical independencies among local ***s.i.m.***. The second term controls the structural complexity (Bayesian Statistical Significance Measure, BSSM, is selected in our work), The last term is to assure that the final fusion network will be on the average more accurate than each local ***s.i.m.*** at each data point. The value of β_{ik} is defined as follows:

$$\beta_{ik} = \begin{cases} 0 & \text{if final fuser outperforms the } i^{th} \text{ local s.i.m. at data point k} \\ 1 & \text{ortherwise} \end{cases} \quad (16)$$

Refer to Fig. (5) again, if we consider a z-transform of each linear filter block as

$$1^{st}\ order : \frac{Y(z)}{U(z)} = \frac{K_1}{1 + a_1 z^{-1}} \quad (17.1)$$

$$2^{nd}\ order : \frac{Y(z)}{U(z)} = \frac{K_2}{1 + a_2 z^{-1} + a_3 z^{-2}} \quad (17.2)$$

There is a chance during GA learning process that some of coefficient a escape into an unstable area ($|a|>1$). So in order to assure that a stable solution of Eq. (17) always exists, we must post one more constraint as:

$$|a_1|, |a_3| \leq 1 \quad (18.1)$$

$$|a_2| < \sqrt{1 + 4a_3} \quad (18.2)$$

It is worth noting that Eq. (17)-(18) generalize the fusion architecture to an unlimited order of the linear filter portion of the fusion network. However, for sake of verifying our idea, we will simply limit the maximum order to five. This leads to a possible combination of one first-order, and two second-order linear filter blocks.

6. EXPERIMENTAL SETUP

6.1 Local *s.i.m.* Construction Process

Fig. (6) shows a sensor fusion test-bed to determine the simulated heading movement of our autonomous flying robot in yaw movement. A set of sensors consists of a piezo-electric angular accelerometers (and gyroscope) model from CFX Technologies. One GY-240 rate gyro model from FUTABA, and TCM2-50 (microprocessor-controlled fluxgate digital

compass from PNI Cooperation). The reference heading output of the test-bed happens to be from a shaft encoder from KOYO with 2500 pulse per round specification (0.144 degree per pulse). In the experiment, the sampling rate of 25 Hz is used to update heading position via a sensor fusion engine. During data gathering process, the test-bed is rotated in both directions at different speed configurations. Regarding of all the sensor models, the digital compass is the poorest one; i.e., while the test-bed is moving with the faster speed, the response of the compass is more prone to error. The fusion process then aims at reducing such an adversary effect, by augmenting the accelerometer, and the rate gyro with the original compass to improve the quality of the heading data.

We split the sensor measurement input vector into 3 subsets; 1:{compass, gyro#1}, 2:{compass, accelerometer}, and 3:{accelerometer, rate gyro}. Each subset of sensors forms one local *s.i.m.* (RBFN), and each local *s.i.m.* is independently trained on 3 different data sets as shown in Fig. (7). As a result, there are totally 9 local *s.i.m.* to be trained. Noting that from Fig. (7), "NETi/j" means local *s.i.m.* with an input $subset_i$, and trained on $data\ set_j$, where i, j =1,2,3. Each "NETi/j" is found based on our HGAOLS algorithm, and the final network combination is formulated via the EEA method.

Regarding to HGAOLS algorithm in Fig. (4), a floating version of GA with population of 30 chromozomes is implemented in our experiment. Each individual chromosome consists of structural and process parameters. A set of structural parameters includes the center location *(C)*, radius of dead zone *(R)*, and the receptive field width of each basis function *(σ)*. While the process parameters involve with crossover (P_c) and mutation probability (P_m)constant. In the experiment, we use decaying functions of P_c and P_m with high initial values (P_c=0.7 and P_m=*0.5*) in order to promote information interchange among parental chromosomes and mutation on individual chromosome. The mutation process is based on a Gaussian distribution, and the crossover between two parental chromosomes is an intermediate combination. An elitism selection process with a ratio of 5-8% is utilized, meaning that the first two fittest chromosomes (out of 30) are preserved for the next generation and the rest are selected based on stochastic uniform technique.

Table (1)-(2), and Fig. (8) give the empirical results on each local *s.i.m.* being trained on 2000 data points. One can easily see that each local *s.i.m.* comes with different set of structural parameters (dead zone radii, receptive field width, and height of each basis function). Also they perform with different levels of accuracy (Table (1)). However, all outperforms the original digital compass in the means square sense. On the experiment, the best local *s.i.m.* is NET2/3 and the second runner is NET2/1. Even though NET2/3 contains more nodes than NET2/1, it is not necessary that it will perform better. This verifies the key idea of our HGAOLS algorithm (Fig. (8)). In fact, it is able to trade-off between complexity (network size) and accuracy (MSE on training data) in order for the final local *s.i.m.* (RBFN) to best perform on an unseen data (minimum generalization error).

6.2 Global Fuser Construction Process

For the global fuser construction process, referring to fig. (5), the goal is to find an optimum set of all constant parameter inside the global fuser engine; i.e., weighting constant for each NETi/j (w_{ij}), an input delay coefficient (b_1, b_2), and all

coefficients contained in each linear filter block (a_i; i=1..5). We create a population of 50 chromozomes. It takes 80 iterations until the algorithm stops. Noting that at this step, a chromosome topology is different from that of local *s.i.m.*. In fact, a set of structural parameters is as $\{w_{ij}, b_1, b_2, a_1, a_2, a_3, a_4, a_5, n_w, n_b, n_a\}$. N_w, n_b, and n_a are on/off switching patterns for local *s.i.m.*, delay unit, and linear filter block. These three constant parameters lead to adaptive fusion network configuration of each candidate chromosome.

Table (3) shows results on the top 4 winners and Fig. (9) gives a performance comparison among the local *s.i.m.*, global fuser, and original compass. Also, we calculate a simple averaging network (NET_{AVG}) by adding an output from all the *local s.i.m.* and divided by the total number of those local *s.i.m.*. One can see that both of the NET_{AVG} and final global fuser engine yield a better performance than even the best local *s.i.m.*, NET2/3. So combining more than one local *s.i.m.* really makes thing work out than using each individual local *s.i.m.*. However, in case of NET_{AVG}, the cost of higher accuracy (MSE) is high complexity of the final network configuration (all basis function nodes are connected). To solve such a problem, our EEA algorithm lets some of the unnecessary local *s.i.m.* to be discarded during the search process making sure that the complexity and accuracy is best trade-off. Regarding Table (3), candidate#1 has higher MSE (17.0721) than cadidate #3 (15.6769). However, when taking a structural complexity into account (described as MSE_{total} ; fitness in Eq. (15)), candidate#1 is the first winner since it contains one local *s.i.m.* less than candidate#3. Consequently, we can conclude that our EEA algorithm provides the best network combination in the sense that all the objectives in Eq. (15) are most satisfied.

7. CONCLUSION

In this paper, we propose a viable approach based on a mixture of divide-and-conquer principle, RBFN, and EA to fuse highly corrupted data from a digital compass with those from a rate gyro and angular accelerometer such that the reliability and robustness on heading information of an autonomous flying robot is greatly improved in the sense that an uncertainty hyper-ellipsoid of the fused data is minimized. To generalize our problem, the whole fusion process is treated as a generic, time-invariant, nonlinear dynamics mapping which relates a sensory (raw) data input vector to a fused data output (heading position). The architecture contains two hierarchical levels: local and global fuser engine; hence decomposing design process into two independent steps. The HGAOLS algorithm is proposed to construct each local *s.i.m.* (RBFN), and EEA is used to search for the final global fuser engine.

The empirical data indicate that our HGAOLS algorithm can exploit a full advantage of the original OLS in an intelligent way to yield a parsimonious local *s.i.m.* (RBFN). In one hand, based on idea of ensemble machines, the EEA can adaptively combine the local *s.i.m.* such that the final global fuser engine can best trade-off between the complexity and the accuracy through the fitness function mechanism of the GA.

Our sensor fusion architecture is also modular; hence one can add a new sensor set as local *s.i.m.* at any time in the future. This can be easily done by initializing the weighting factor of a newly trained local *s.i.m.* to zero, and have the EEA learn a new global network configuration. Further, the resulting fusion engine is robust to uncertainty of sensor

statistics. This is verified by our experimental results at different speed configurations that our global fuser outperforms the best local *s.i.m.*. Lastly, the fusion architecture is considered as a generic model for any type of fusion, meaning that a designer has freedom to select any kind of nonlinear mapping part and the linear dynamic filter block with various orders to satisfy his design objective with a reasonable level of accuracy. Regardless of the type of mapping and order of the filer, our HGAOLS and EEA algorithms can still be applied to tackle the new problem.

Acknowledgement

This research project is financial supported by Thailand Research Fund.

References

[1] Alhakeem, S. and Vashney, P.K. (1996.), Decentralized Bayesian Detection with Feedback, *IEEE Trans. On Systems, Man, and Cybernetics*, Vol. 26, No. 4, pp. 503-513.

[2] Bar-shalom, Y.B. Li, X.R. and Kirubarajan, T. (2001), *Estimation with Applications to Tracking and Navigation*, John Wiley & Son, Inc.

[3] Blanziera, E. (2003), "Theoretical Interpretation and Application of Radial Basis Function Networks", Technical Report#DIT-03-023, University of Trento, Italy, submitted to Elsevier Science, May.

[4] Chen, S. Cowan, C.F.N. and Grant, P.M. (1991), "Orthogonal Least Squares Learning Algorithm For Radial Basis Function Networks", IEEE Trans. Of Neural Networks, Vol.2, No.2, March.

[5] Dietterich, T.G. (2000), *Ensemble Methods in Machine Learning*, In Kittler, J. and Roli, F., editors, *Multiple Classifer Systems*, Vol. 1857 of lecture Notes in Computer Science, pp. 1-15, Cagliara, Italy.

[6] Hashem, S. (1994), "Optimal Combinations of Neural Networks", *Neural Networks*, Oct.

[7] Haykin, S. (1999), *Neural Network: A Comprehensive Foundation 2nd edition*, Prentice Hall Inc., Upper Saddle River, New Jersey, N.W.

[8] Hiransoog, C. and Malcolm, C.A. (1999), "Multi-sensor/ Knowledge Fusion", *Proc. of the 1999 IEEE Int. Con.On Multisensor Fusion and Integration for Intelligent Systems*, Taipei, Taiwan, pp.117-122, Aug (2):285-299.

[9] Kamberova, G, Mandelbaum, R. and Mintz, M. (1996), "Statistical Decision theory for Mobile Robotics: Theory and Application", in Proc. Of 1996 IEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems (Washington DC, Dec.). pp. 17-24.

[10] Kobayashi, F. Arai, F Fukuda, T. Onoda, M. and Hotta, Y. (1999), "Sensor Selected Fusion System (Application to Inference of Surface Roughness in Grinding System)", Proc. of the 1999 IEEE Int.Con.On Multisensor Fusion and Integration for Intelligent Systems, Taipei, Taiwan, August.

[11] Kobayashi, F. Arai, F. Shimojima, K. and Fukuda, T. (2000), *Sensor Fusion System Using Recurrent Fuzzy Inference*, In Sinha, N.K. Gupta, M.M., editors, *Soft Computing and Intelligent Systems: Theory and Application*, Academic Press, A Harcourt Science and Technology Company, San Diego, CA, USA.

[12] Lefebvre, T. Bruyninckx, H and De Schutter, J. (2001), "Kalman Filters for nonlinear systems: a

comparison of performance", Internal Report#01R033, Department of Mechanical Engineering, Katholieke Universiteit, Leuven, Belgium, Oct.

[13] Ljung, L. (1999), System Identification: Theory for theUser, Prentice Hall PTR, Upper Saddle River, New Jersey.

[14] Lua, R.C. and Su, K.L. (1999), "A Review of High-Level Multi sensor Fusion: Approaches and Applications", Proc. of the 1999 IEEE Int. Con. On Multi-Sensor Fusion and Integration for Intelligent Systems, Taipei, Taiwan.

[15] Majumder, S. (2001), "Sensor fusion and Feature Based Navigation for Subsea Robot", Phd. Thesis, University of Sydney.

[16] Michalewicz, Z. (1999), *Genetic Algorithms + Data Structures = Evolution Programs: Thrid Edition*, Springer Verlag, printed in USA.

[17] Niu, S.S. and Ljung, L. (1994), "Multiple Model Parameter estimation", Technical Report, Department of Electrical Engineering, Linkoping University, S-58183 Linkoping, Sweden.

[18] Perrone, M.P. (1993), "Improving Regression Estimation: Averaging Methods for Variance with Extensions to General Convex Measure Optimization", Phd. thesis, Brown University.

[19] Raftery, A.E. Madigan, D. and Hoeting, J.A. (1997), "Bayesian Model Averaging for Linear Regression Model", Jounal of the American Statistical Association (1997) 2, pp. 179-191.

[20] Roumeliotis, S.I. and Bakey, G.A. (1997), "An Extended Kalman Filter for Frequent Local and Infrequent Global Sensor Data Fusion", In SPIE International Symposium on Intelligent Sysems and Advanced Manufacutring.

[21] Thrun, S. Fox, D. and Burgard, W. (1997), "Probabilistic Methods for State Estimation in Robotics", Proc. Of The Workshop SOAVE'97, VDI-Verlag, pp. 195-202.

[22] Tresp, V., *Committee Machines*, In Hu, Y.H. and Hwang, J.N. (2001), editors, *Handbook for Neural Network Signal Processing*, CRC Press.

[23] Wan, E.A. and Van Der Merwe, R. (2000), "The Unscented Kalman Filter for Nonlinear Estimation", In Proc. Of IEEE Symposium (AS-SPCC), Lake Louise, Alberta, Canada, Oct.

[24] Wijesoma, W.S. Khaw, P.P. and Teoh, E.K. (2001), "Sensor Modelling and Fusion for Fuzzy Navigation of an AGV", Int. J. of Robotics and Automation, Vol. 16, No.1.

[25] Wu, H. Siegel, M. Stiefelhagen, R. and Yand, J. (2002), "Sensor Fusion Using Dempster-Shafer Theory", IEEE Instrumentation and Measurement Technology Conference, Achorage, AK, USA, 21-23 May.

[26] Zhou, Z.H. Wu, J.X. Jiang, Y. and Chen, S.F. (2001), "Genetic Algorithm based Selective Neural Network Ensemble", in Proc. of the 17th Int. Joint Conf. on Artificial Intelligence, Seattle, WA, vol.2, pp.797-802.

Figure1 An autonomous flying robot platform under studied. The robot is equipped with different kinds of sensors to detect its movement

Figure 2 Sensor Fusion Architecture. (a) The sensor fusion engine is treated as a generic nonlinear, time-invariant, dynamics module; relating a set of (raw) sensory input to an out put as a fused data. (b) A so-called "Hammerstien" model is implemented as our proposed fusion engine. It consists of a series connection of nonlinear static mapping (local fuser) and linear dynamics model (global fuser)

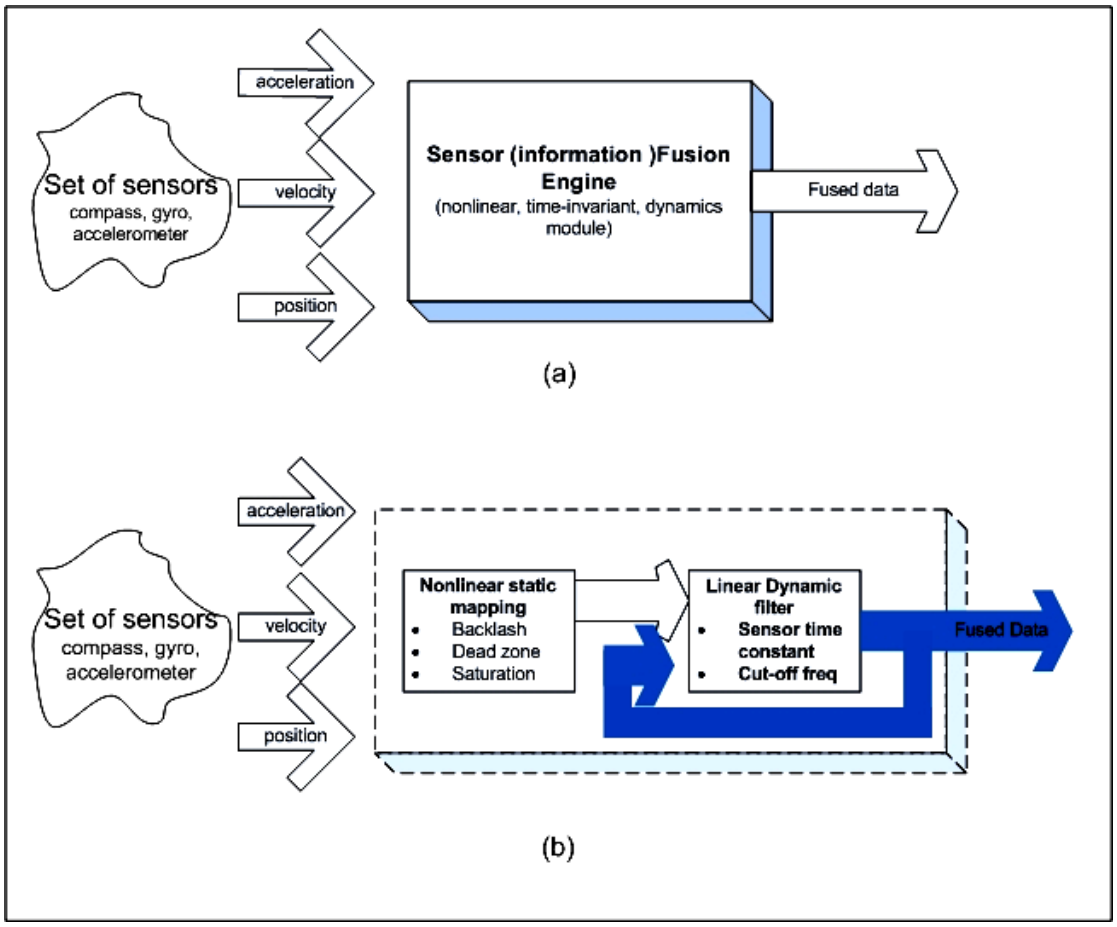


Figure 3 Inside the sensor fusion network, each individual local fuser engine is modeled by RBFN, while a global fuser engine is represented by a linear filter model (ARX model)

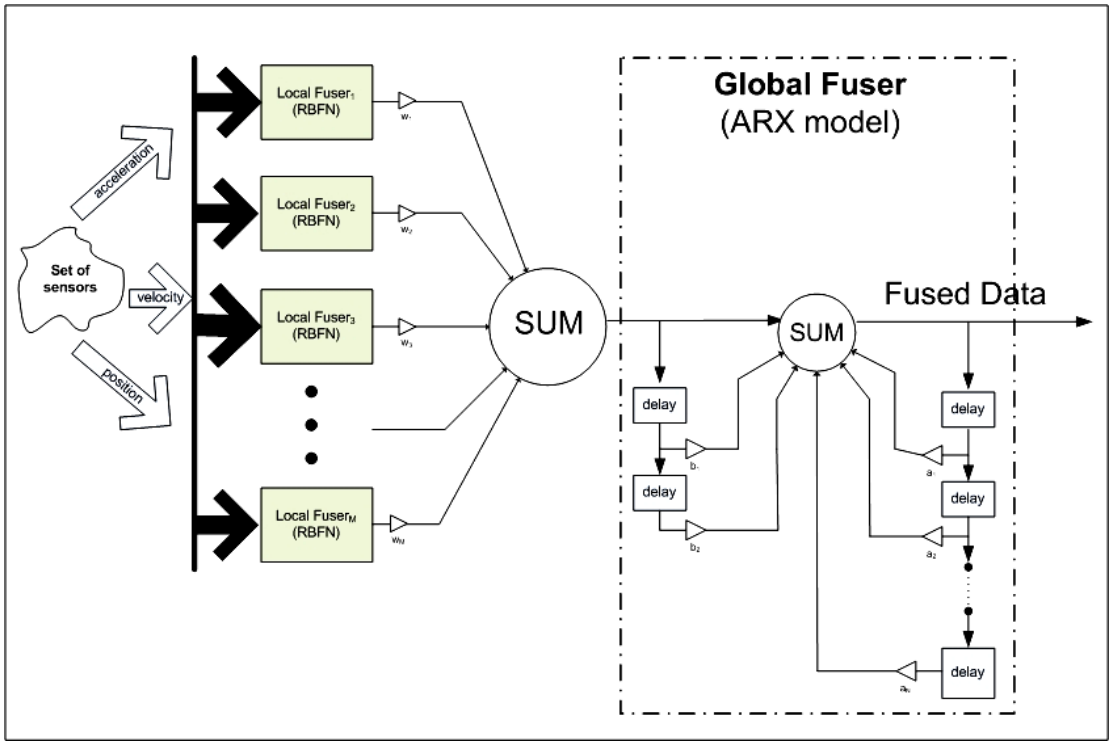


Figure 4 Hybrid ES/OLS algorithm is used in training an individual local fuser engine (RBFN). It is a modified OLS algorithm with an adaptive structure on each basis function of the network

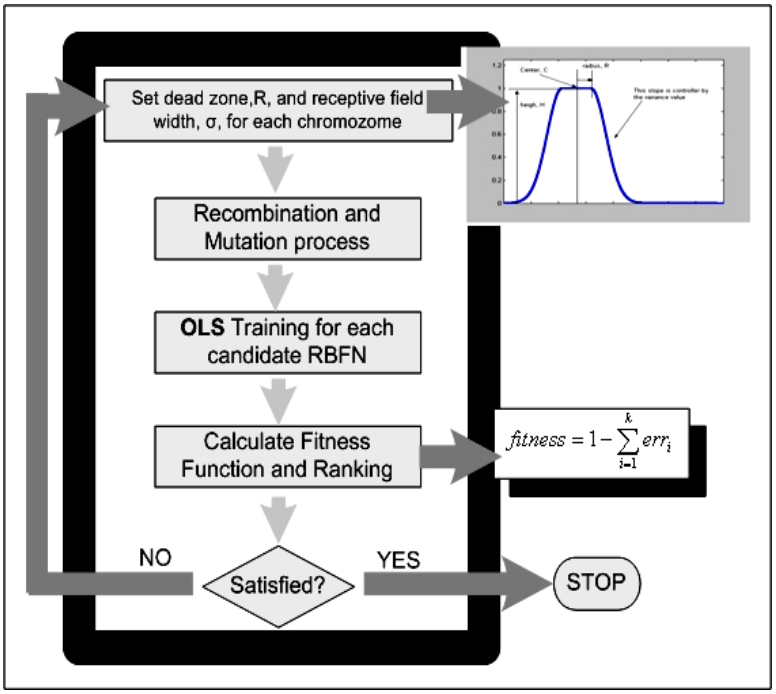


Figure 5 EEA algorithm allows an adaptive structure of sensor fusion network. Its ultimate goal is to find the best possible solution on both a weighted linear combination of a subset of local fuser networks, and the final global fuser topology

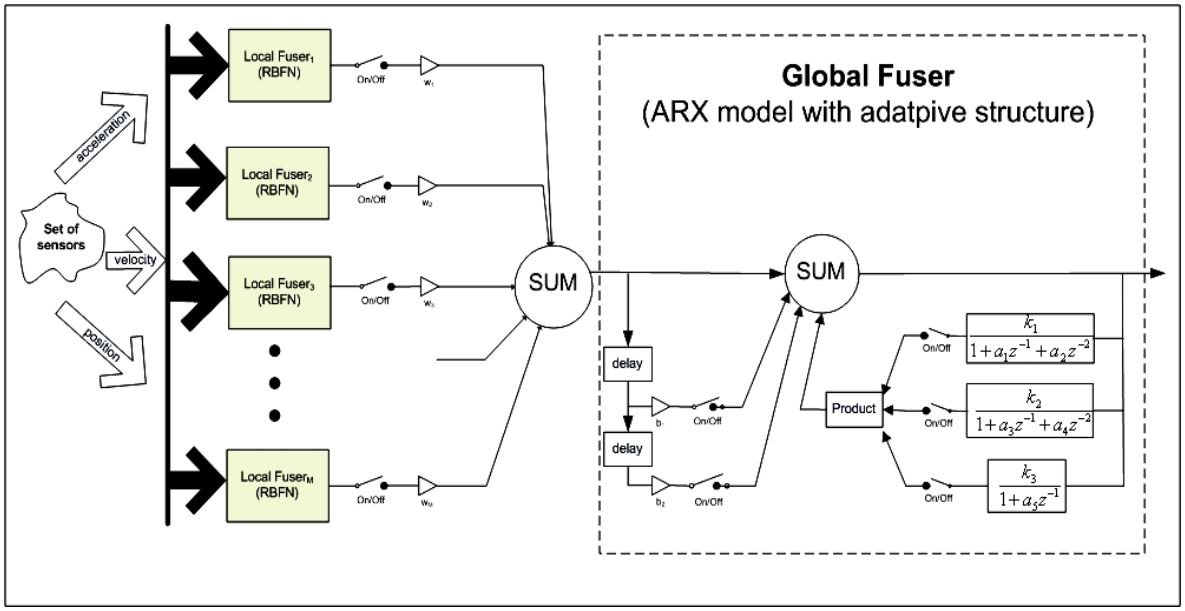


Figure 6 Test Bed for hardware simulation on sensor fusion network for our flying robot to determine its heading information

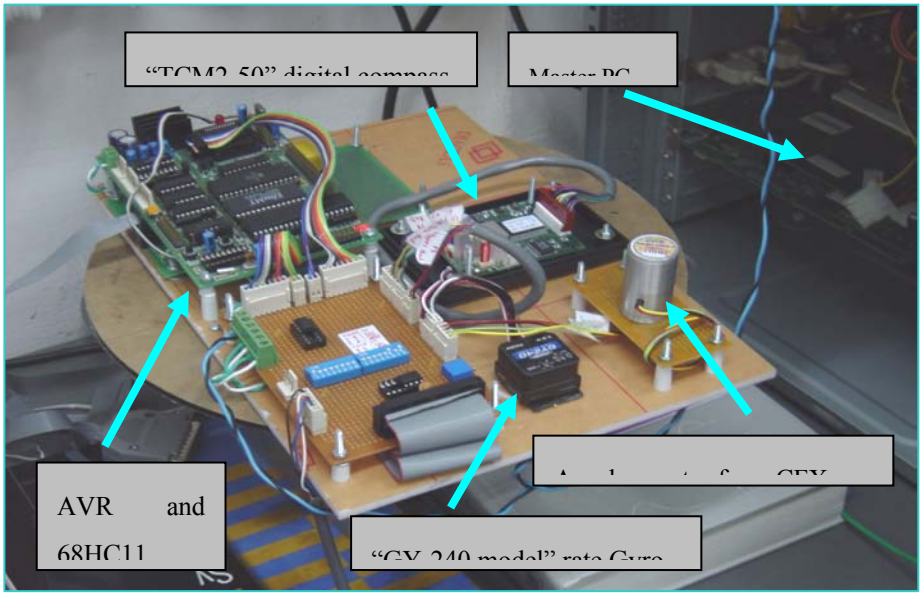


Figure 7 The training data are collected from experiment under different conditions. To train each local fuser, the whole data pool is split into three data sets, each of which is randomly selected from the same distribution. Also, each local fuser is fed with a subset of input vector.

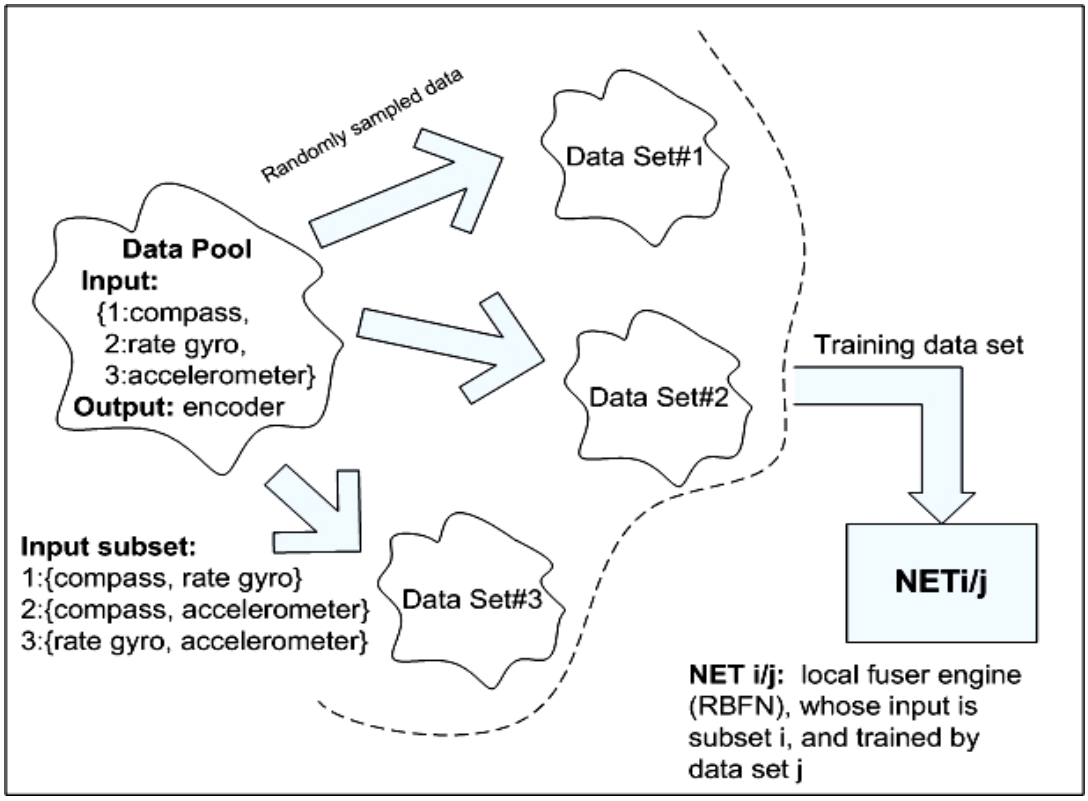


Figure 8 All local fuser engines (RBFN) are independently trained by Hybrid ES/OLS algorithm. The best local fuser is the one which best trade-off the complexity (number of nodes) and accuracy (lower MSE) compared with the real data

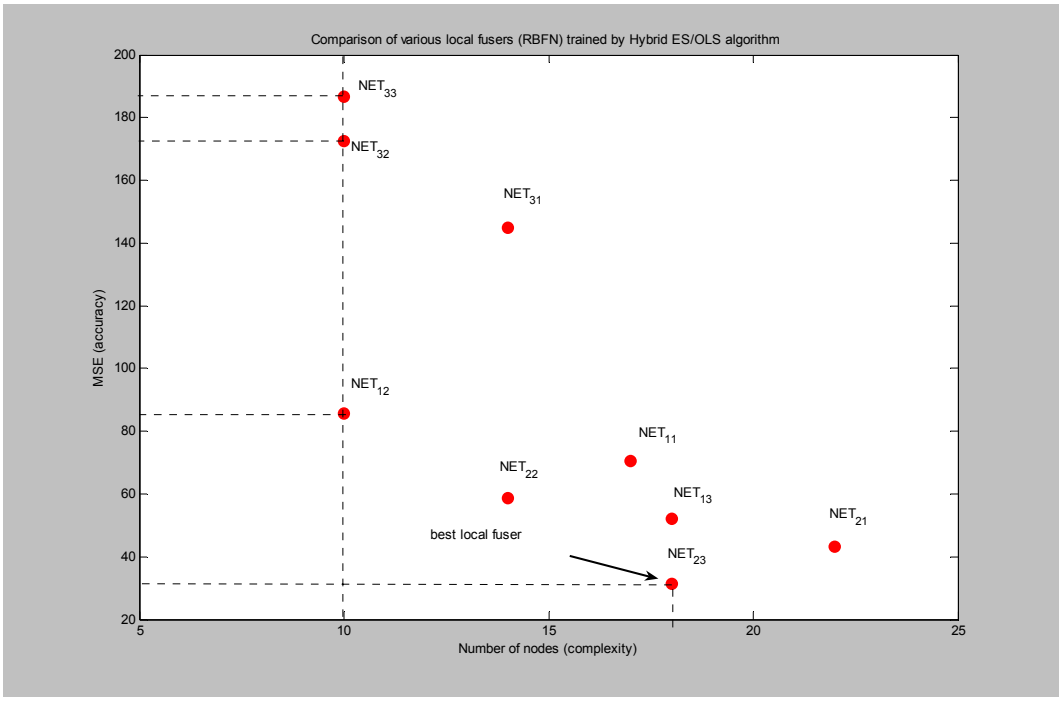


Table1: Resulting Local Fusers compared with original digital compass on Test Set [2000 data points]

Module	MSE	Num of Node	Rel. MSE	
compass	**484.7365849**	-	**0.364485692**	
Net1/1	70.63411128	17	0.053111574	
Net1/2	85.52948096	10	0.064311779	
Net1/3	51.93115431	18	0.039048348	
Net2/1	**43.1681132**	22	**0.032459196**	← Second runner
Net2/2	58.64328534	14	0.044095369	
Net2/3	**31.30426693**	18	**0.02353847**	← Best local fuser
Net3/1	144.7417427	14	0.108834975	
Net3/2	172.6081981	10	0.129788468	
Net3/3	186.6224398	10	0.14032613	

Table2: Local Fuser configuration. Each basis function has two input elements [physical sensors].

Module	Num of Node	Basis Function Configuration					
		dead zone radius (r)		height (h)		receptive field width (σ)	
Net1/1	17	[99.47627	99.51393]	[0.45765	0.25997]	[63.84345	63.78425]
Net1/2	10	[130.30891	119.97139]	[0.48874	0.91872]	[90.23078	90.59792]
Net1/3	18	[155.07836	106.75874]	[0.43982	0.63708]	[0.43982	0.63708]
Net2/1	22	[154.40800	125.71423]	[0.40692	0.57588]	[100.000	100.000]
Net2/2	14	[155.40772	107.77353]	[0.24711	0.69409]	[100.00000	77.96618]
Net2/3	18	[0.00000	0.00000]	[0.20000	0.20000]	[93.58083	93.51969]
Net3/1	14	[142.19408	150.93108]	[0.48840	0.51848]	[56.82354	48.34572]
Net3/2	10	[0.00000	18.60694]	[0.20000	0.97978]	[89.00272	100.0000]
Net3/3	10	[153.3542	0.00000]	[0.7852	1.00000]	[101.1358	51.56257]

Table 3 Resulting configurations of the candidate sensor fusion architectures from the EEA algorithm [num of pop: 50, iteration: 80]

Candidate # 1:

w_{11} = 0.1766 w_{12}= 0.0633 w_{13}= 0.2079

w_{21} = 0.0000 w_{22}= 0.0000 w_{23}= 0.2633

w_{31} = 0.0000 w_{32}= 0.1599 w_{33}= 0.0000

b_1= 0.5507 b_2=0.2751

a_1= 0.3943 a_2= 0.0000 a_3= 0.6203 a_4= 0.0000 a_5 = 0.0000

MSE = 17.0721 MSE_{total} = **1263.3381**

Candidate # 2:

w_{11} = 0.2298 w_{12}= 0.0835 w_{13}= 0.1734

w_{21} = 0.0000 w_{22}= 0.0000 w_{23}= 0.2503

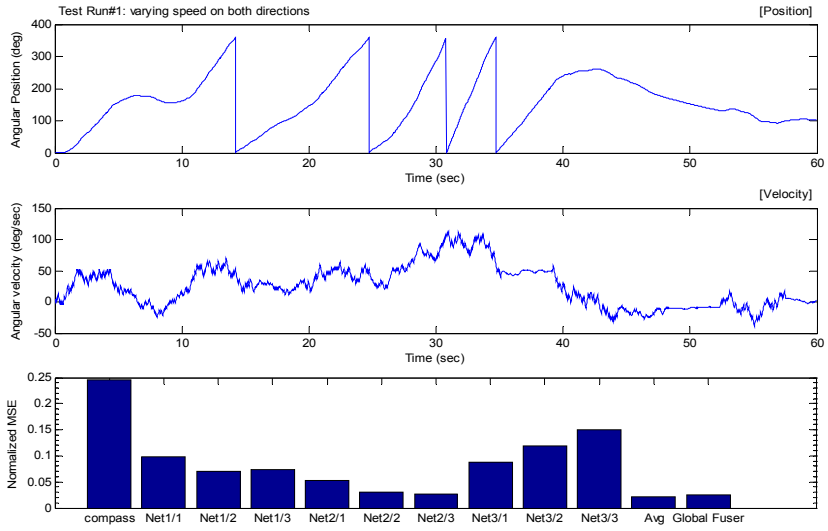
w_{31} = 0.0000 w_{32}= 0.2334 w_{33}= 0.0000

b_1= 0.4510 b_2= 0.0000

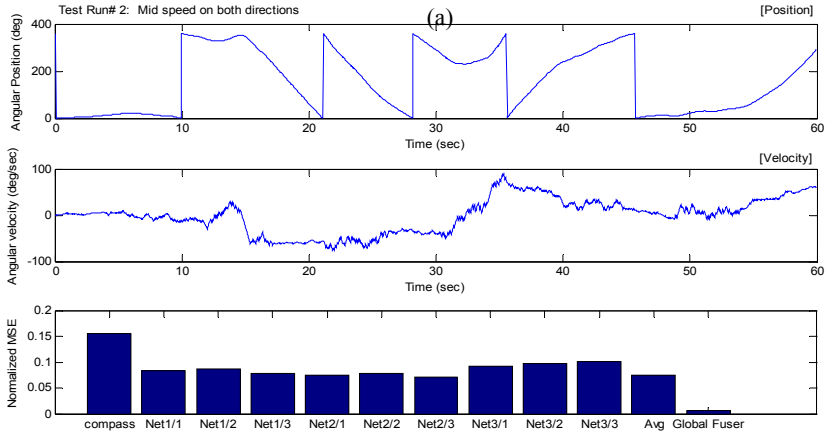
a_1= 0.5464 a_2= 0.0000 a_3= 0.3687 a_4= 0.0000 a_5 = 0.0346

MSE = 17.3100 MSE_{total} = **1280.9375**

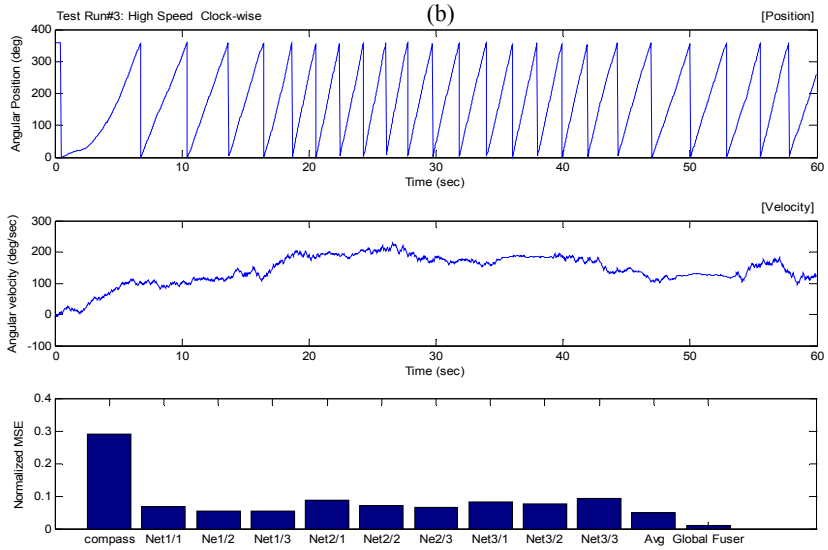
Figure 9 The performance of a final sensor fusion engine compared with an original digital compass, each individual local fuser, and a simple averaging of all local fuser networks



(a)



(b)



(c)

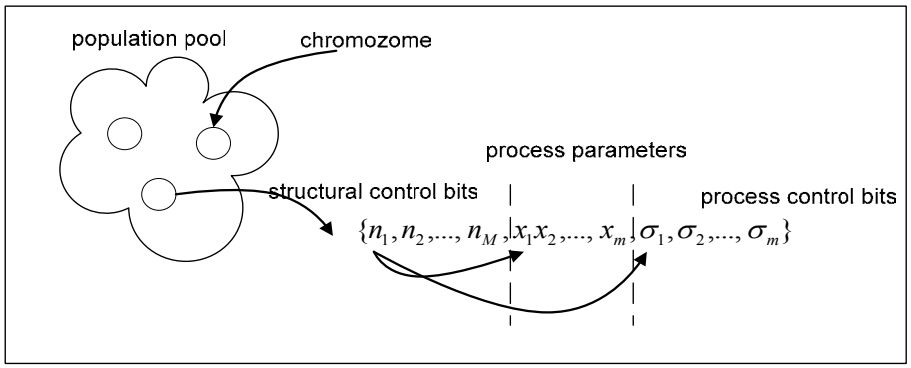


Figure 7: The multi-level structure of chromosome in HES.

DEVELOPMENT AND CONTROL OF 6 DOF FULLY AUTONOMOUS FLYING ROBOT BY HYBRID ADAPTIVE NEURO-FUZZY MODEL REFERENCE CONTROL

SUKON PUNTUNAN and MANUKID PARNICHKUN
Asian Institute of Technology, P.O. Box 4, Klong Luang, Pathumthani, 12120, Thailand
Tel: +66-2-524-5229, Fax: +66-2-524-5697, Email: manukid@ait.ac.th

Abstract

Control of 6-DOF fully autonomous helicopter type flying robot is very difficult. Many researchers verified their control algorithms only on simulation. There are very few success experiments on fully control of the flying robot. In order to make the robot fly autonomously, the attitude and position controls are needed. In this paper, the neuro-fuzzy controllers (NFC) are developed to control the roll, pitch and yaw of the flying robot, while the hybrid adaptive neuro-fuzzy model reference control (Hybrid-ANFMRC) is developed to control its position. The attitude controllers are trained offline to zero out the roll, pitch and yaw errors. The position control uses the hybrid technique called, "hybrid adaptive neuro-fuzzy model reference control". The position control learns online to track the velocity reference model, while trying to obtain the smooth response and zero steady state error. Design robustness of the proposed control algorithm is addressed by testing in the experiments under various ranges of the controller gains. In this paper, the experimented results are used to show the performance of the proposed control algorithm.

Keyword: neuro-fuzzy control, flying robot, adaptive control, hybrid control, model reference adaptive control.

1. Introduction

Over the coming century, flying robots will take the place of human labor in many areas, particularly in various hazardous duties. For example, they can hover and transmit video image from hostage situations, enemy positions, or areas contaminated by chemicals or biological agents. To make use of robots in these various circumstances, they should have the ability to fly automatically. A flying robot developed at AIT is modified from X-Cell 60 radio-controlled helicopter. It is developed to support autonomous flight control covering wide-mode missions of operation from hovering to other maneuvers. The flying robot has six degrees of freedom in its motion. It can make various flights, such as forward flight, backward flight, sideward flight, hovering, vertical climb, etc. The problem with this kind of flying robot is that it is inherently unstable, especially at low speed. There are nonlinear variations in the dynamics with air speed. Also the natural environment such as wind easily affects the flight dynamics. Hence control of the robot is a difficult one.

Currently, there are many researches on development of autonomous flying robots with different control techniques [7]. There are very few success experiments on fully control of this kind of robot. Two groups of researcher can be considering separately. The first is the researcher who is related on the model-based control. The second is the researcher who is concentrated on the model free approach. The first way is very difficult to make it usable in the real world, because it is difficult to find the acceptable accurate dynamics model. As system increase in complexity, completely and accurately deriving their mathematical models become more difficult. Therefore, the equations that model a system are approximations. To overcome this drawback, some recent research projects have scope to the model free designed technique. Neural network and fuzzy logic are the most popular controllers that have been used. In [2], the neural network controller is trained offline from the flight data. Its uses direct mapping of sensor inputs to the actuator. The control used a "cause" and "effect" approach. Their experiment result is not accomplished with this approach. In [3], a "teaching by showing" methodology is developed to train the fuzzy-neural controller. The controller is generated and tuned using training data gathered while the teacher operates the flying robot. The methodology has been successfully applied in simulation but failed to control the flying robot for real world validation. In [5], a fuzzy logic controller was successfully applied to control the flying robot. Their used the knowledge and technique of an experienced pilot/engineer to design the fuzzy logic controller. Their also compared the performance of fuzzy logic control and linear control under a windy environment. Fuzzy control shows much more robustness against winds than linear control. However, the designed process used much more time and required the experimental skill from the expert pilot. The drawback of the fuzzy logic is that it requires more knowledge about the operation of the plant. Normally, the parameters of the fuzzy logic controller need to be finding manually. The drawback of the neural network controller is that it is difficult to re-tune it after the training process is accomplished. The neuro-fuzzy controller combines the advantage of the fuzzy logic controller and neural network together. The learning capability of the neural network and the tuning capability of the fuzzy logic controller are merged.

In this paper, the model free approach is developed. The neuro-fuzzy is proposed to control the roll, pitch and yaw of

the flying robot. The neuro-fuzzy is trained from the flight data. The hybrid adaptive neuro-fuzzy model reference control (Hybrid-ANFMRC) is proposed to control the position of the flying robot. The position control combines the neuro-fuzzy with the proportional control. The proportional control does as the basis controller, while the adaptive neuro-fuzzy model reference control try to learn to track the velocity reference model. The reference model is defined as the function of the position error. It can be the linear or exponentially relation of the position error. The position control learns from scratch without using any expert knowledge. Experiments were undertaken to evaluate the efficiency of the proposed control algorithm. The robustness of the position controllers was addressed by testing in the experiments under various ranges of the proportional gains.

This paper is organized as follows. In section 2, provides a description of the flying robot. Section 3 provides the structure of the neuro-fuzzy and the hybrid adaptive neuro-fuzzy model reference control. Section 4 provides the simulation and experimental results, which demonstrate performances of the proposed control algorithm. Finally, the conclusion is made in section 5.

2. System Description

The flying robot airframe is a modified X-Cell 60 radio-controlled helicopter with a main rotor diameter of 1.80 meters. The robot's OS91 glow plug engine has power rating of 3.0 HP, resulting in the maximum payload of 5.0 kg and flight duration of approximately 15 minutes. Fig. 1 shows the flying robot and its avionics box. The control system contains the following processors and sensors.

- The flight control microprocessor, based on the 16-bits digital signal controller. The flight control microprocessor controls the roll, pitch, yaw and position of the flying robot. It is also generated pulse width modulation (PWM) signals to drive 4 actuators. The flight control microprocessor continuously communicates with ground station via a serial radio modem. The communication occurs every 200 milliseconds and the range of communication covers up to 3 km. The ground station sends DGPS correction signal and updates user commands to the flying robot.
- A 3DM-GX1 attitude and heading reference sensor (AHRS) containing three angular rate gyros, three orthogonal linear accelerometers, and three orthogonal magnetometers to provide three orientation angles (roll, pitch and yaw).
- An OEM4 RT-20 GPS card. The GPS provides latitudes and longitudes information within 20 cm CEP (circular error probable) when operated in a real time kinematics.
- There are 2 altitude sensors. In a short range and high precision measurement, an SRF-08 ultrasonic altimeter is used to provide ground-to-robot distance. At the higher altitude, the barometric pressure altimeter is used to provide the altitude with 1-meter resolution.

Fig. 1. Flying robot testbed

The robot has five control inputs [5].

- The throttle, δ_{Thr} this is the input for the engine control to drive the rotor. There is a feedback loop for the throttle control to maintain the rotation speed of the main rotor constant. The engine governor is used for this purpose.
- The collective pitch, δ_{Col} this is the input for the climb or descent control by changing the main rotor's lift though the change of the main rotor blade angle. It is used in the altitude control.
- The longitudinal cyclic or elevator, δ_{Lon} this is an input for forward and backward flight control by tilting the main rotor path plane forward or backward. It is used in the pitch and longitudinal position control.
- The lateral cyclic pitch or aileron, δ_{Lat} this is the input for the rightward or leftward flight control by tilting the main rotor path plane right or left. It is used in the roll and lateral position control.
- The rudder cyclic pitch, δ_{rud} this is an input for the yaw control by changing the lift of the tail rotor through the change of the tail rotor blade angle. It is used in the yaw control.

Most of the researchers [7] have used the PC104 computer as the onboard computer. In our flying robot, the onboard computer is an embedded microprocessor as shown in Fig. 2. For the small-sized flying robot, this can be reduced the weight, space and electrical power consumption of the control system. The control cycle of one completely calculation must be within 20 microseconds (50 Hz). The control algorithms presented in this paper is not only designed in the control performance viewpoint. It is optimized to suit with the low computing power of the embedded microprocessor. So, all of the membership function in the fuzzy layer of the neuro-fuzzy and the Hybrid-ANFMRC are selected as the symmetrical triangle membership functions.

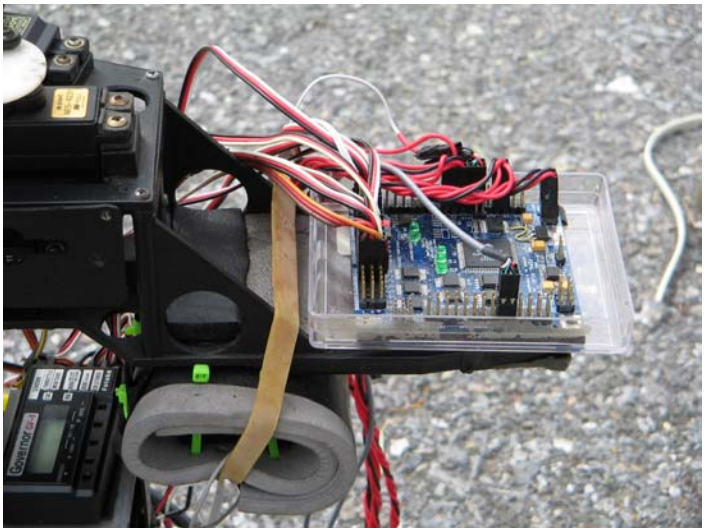


Fig. 2. The onboard microprocessor

3. Control Algorithms

3.1 Neuro-fuzzy control

The neuro-fuzzy is developed to control the roll, pitch and yaw of the flying robot. The neuro-fuzzy controllers constitute a class of hybrid soft controllers that fuse fuzzy logic and neural networks. It combines the advantages of neural network in learning ability, optimization abilities and connectionist structure with the advantage of fuzzy logic control in human like structure, ease of incorporating expert knowledge [1]. The structure of the neuro-fuzzy attitude controller is shown in the Fig. 3.

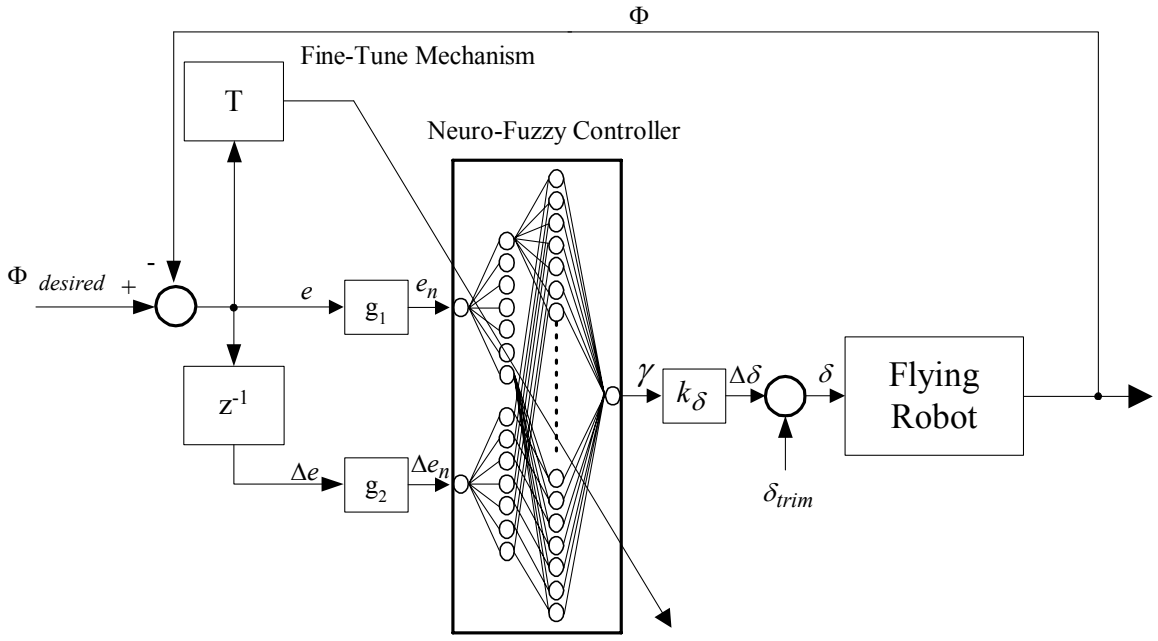


Fig. 3. Neuro-fuzzy attitude control

In Fig.3, there are two inputs and one output of the neuro-fuzzy control. The first is the attitude error, $e(k)$. The second is the change of attitude error, $\Delta e(k)$. The output of the neuro-fuzzy is the change of the actuator command, $\Delta\delta(k)$. The attitude error, $e(k)$ and change of attitude error, $\Delta e(k)$, are determined as followings.

$$e(k) = \Phi_{desired}(k) - \Phi(k) \tag{1}$$

$$\Delta e(k) = e(k) - e(k-1) \tag{2}$$

where $\Phi_{desired}(k)$ is the desired attitude, and $\Phi(k)$ is the actual attitude of the flying robot.

The input variables of the neuro-fuzzy are normalized to the normalized attitude error, $e_n(k)$ and the normalized change of attitude error, $\Delta e_n(k)$. The normalized values are calculated as followings.

$$e_n(k) = g_1(e(k)) \tag{3}$$

$$\Delta e_n(k) = g_2(\Delta e(k)) \tag{4}$$

where $g_1(\bullet)$ and $g_2(\bullet)$ are the normalization functions of the attitude error, $e(k)$ and change of attitude error, $\Delta e(k)$, respectively.

The normalization functions are defined as the followings.

$$\begin{aligned} g_1(e(k)) &= k_e g_{1neg} e(k) && if \quad e(k) \le 0 \\ &= k_e g_{1pos} e(k) && if \quad e(k) > 0 \\ g_2(\Delta e(k)) &= k_{\Delta e} g_{2neg} \Delta e(k) \quad if && \Delta e(k) \le 0 \\ &= k_{\Delta e} g_{2pos} \Delta e(k) && if \quad \Delta e(k) > 0 \end{aligned} \tag{5}$$

where k_e and $k_{\Delta e}$ are the attitude error and the change of attitude error gains, respectively. The constant values of g_{1neg}, g_{1pos}, g_{2neg} and g_{2pos} are the normalization factors for each input variables.

The output of the neuro-fuzzy control is the result of mapping from the normalized attitude error, $e_n(k)$ and normalized change of attitude error, $\Delta e_n(k)$ to the output, $\gamma(k)$. The change of actuator command, $\Delta\delta(k)$, is obtained by multiplying the output, $\gamma(k)$ with the output gain k_δ as the following.

$$\Delta\delta(k) = k_\delta \gamma(k) \tag{6}$$

The actuator command, $\delta(k)$, is the summation between the change of actuator command, $\Delta\delta(k)$ and the control

trim, δ_{trim} . The value is calculated as the following.

$$\delta(k) = \delta_{trim} + \Delta\delta(k) \tag{7}$$

In addition, the performance of the neuro-fuzzy control is affected by the input-output normalization factors [6]. The normalized values are affected directly by changing the attitude error gain, k_e , the change of attitude error gain, $k_{\Delta e}$ and the output gain k_δ . So, the system time response can be also improved by using the variation of these gains. Fig. 4 is the general system response of the step input.

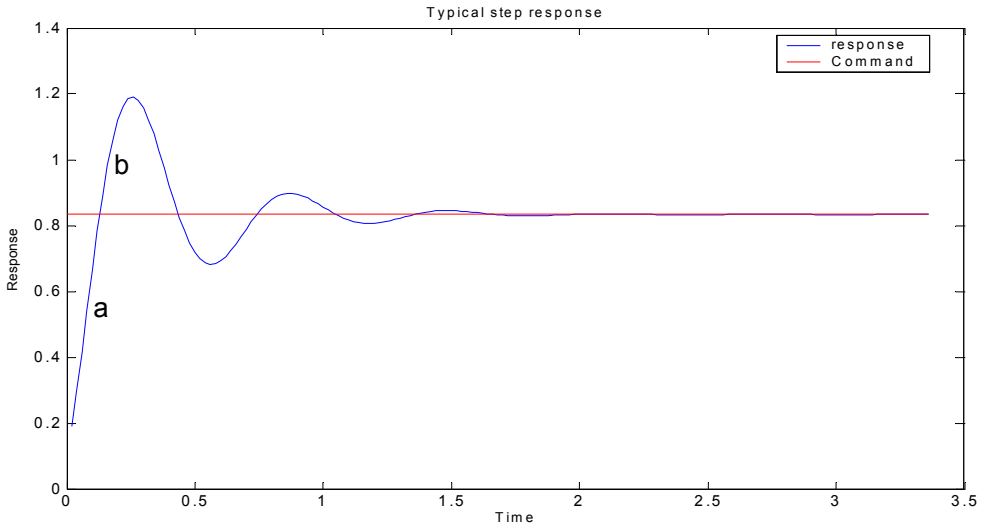


Fig. 4. Typical time response of the step input

For example, the region "a" in Fig. 4, it indicates the system control signal is too small. By increasing of the attitude error gain, k_e and decrease the change of attitude gain, $k_{\Delta e}$, it can improve positive value of control signal and make the response tracking the reference input more quickly. In the region "b", the overshoot happens. One can increase change of attitude error gain, $k_{\Delta e}$. This makes the value of the control effort more negative and reduces the overshoot. This principle is used to re-tune the neuro-fuzzy control after the offline-training is accomplished.

In this paper, the neuro-fuzzy control is trained to zero out the attitude errors. The flight data is used as the training set. The offline training of the neuro-fuzzy control is the back propagation algorithm. In Fig. 5, a neuro-fuzzy control with fuzzy singleton rule is presented. The symmetrical triangle membership function is selected because of its simplicity.

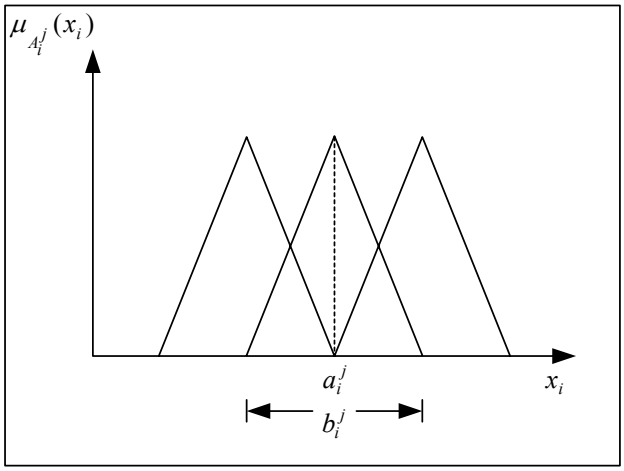


Fig. 5. Symmetrical triangle membership function

The triangle membership function is expressed by the following equitation.

$$\mu_{A_i^j}(x_i) = 1 - \frac{2\left|x_i - a_i^j\right|}{b_i^j},\ i = 1,2,...n\ \ j = 1,2,...,m \tag{8}$$

where x_i is the input value, a_i^j is the center of triangle and b_i^j is the width of triangle. The fuzzy rules, also called fuzzy singleton, are in the following form [1].

$$\textit{Rule } j\textit{: If } x_i \textit{ is } A_1^j \textit{ and } x_2 \textit{ is } A_2^j \text{ and } \ldots \text{ and } x_n \text{ is } A_n^j \textit{ then } \gamma \textit{ is } w_j .$$

where A_i^j is a linguistic term with the membership function, $\mu_{A_i^j}^i(x_i)$, w_j is a real number of weight in the neural part. By the singleton rule, control output, $\gamma(k)$ from the neuro-fuzzy controller is calculated by the following equation.

$$\gamma(k) = \frac{\sum_{j=1}^{m} \mu_j(k) w_j(k)}{\sum_{j=1}^{m} \mu_j(k)} \tag{9}$$

where

$$\mu_j = \mu_{A_j^1}(x_i)\mu_{A_2^1}(x_i)\ldots \qquad \mu_{A_n^1}(x_i) \tag{10}$$

The weights of the neuro-fuzzy control are modified with the steepest gradient method by trying to minimize a cost function. The cost function is defined as the square of the difference between the command attitude and the actual attitude as expressed by the following.

$$E = \frac{1}{2}(\Phi_{desired} - \Phi)^2 \tag{11}$$

The weights of the neuro-fuzzy control are modified with the steepest gradient method as the following.

$$w_j(k+1) = w_j(k) - \eta \frac{\partial E}{\partial w_j} \tag{12}$$

where $\eta \geq 0$ is the learning rate.

By using the chain rule, the adjusted weights can be expressed and calculated as the following.

$$\frac{\partial E}{\partial w_j} = \frac{\mu_j(k)}{\sum_{j=1}^{m} \mu_j(k)} (\Phi_{desired}(k) - \Phi(k)) \tag{13}$$

3.2 Hybrid adaptive neuro-fuzzy model reference control

The Hybrid-ANFMRC is proposed to control position of the flying robot. The control is a hybrid of the proportional control and the adaptive neuro-fuzzy model reference control. In the proposed control algorithm, the proportional control generates the output proportional to the position error. The adaptive neuro-fuzzy model reference control generates the output by learning to track the velocity reference model. The structure of the Hybrid-ANFMRC is shown in Fig. 6.

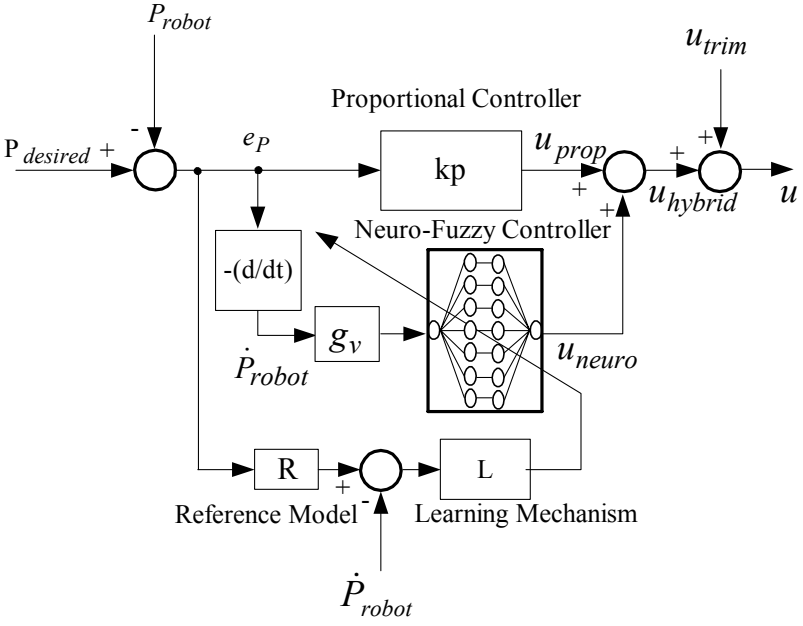


Fig. 6. Structure of the Hybrid-ANFMRC

In Fig. 6, the Hybrid-ANFMRC consists of the proportional control and the neuro-fuzzy control. The position error $e_p(k)$ is the different between the desired position, $P_{desired}(k)$ and the actual robot position, $P_{robot}(k)$. It is calculated as the following.

$$e_p(k) = P_{desired}(k) - P_{robot}(k) \tag{14}$$

The proportional control is used to generate the control output, $u_{prop}(k)$ proportional to the position error, $e_p(k)$. The output of the proportional control is calculated as the following.

$$u_{prop}(k) = u_{prop}(k-1) + k_P(e_p(k) - e_p(k-1)) \tag{15}$$

where k_P is the proportional gain.

The effect of the proportional control will tend to reduce the overall error. However, the effect of the proportional control will reduce as the error approaches zero. In most system, the error will get very close to zero, but will not converge. The adaptive neuro-fuzzy model reference control is used to drive the steady state error to zero, while damp out any oscillation. The output of the proportional control, $u_{prop}(k)$, is summed with the output of adaptive neuro-fuzzy model reference control, $u_{neuro}(k)$ to the hybrid control output, $u_{hybrid}(k)$ as the following.

$$u_{hybrid}(k) = u_{prop}(k) + u_{neuro}(k) \tag{16}$$

The hybrid output, $u_{hybrid}(k)$, is added with the control trim, u_{trim}, to generate the control output, $u(k)$. The calculation is expressed as the following.

$$u(k) = u_{hybrid}(k) + u_{trim} \tag{17}$$

The input of the adaptive neuro-fuzzy model reference control is the velocity of the robot. The robot velocity is normalized to the normalized velocity, $\dot{P}_{n,robot}(k)$. The normalization processes by multiplying the velocity, $\dot{P}_{robot}(k)$ with the scaling factor, g_v as the following.

$$\dot{P}_{n,robot}(k) = g_v \dot{P}_{robot}(k) \tag{18}$$

The output of the adaptive neuro-fuzzy model reference control is the mapping result from the velocity, $\dot{P}_{robot}(k)$ to the adaptable output, $u_{neuro}(k)$. The output of adaptive neuro-fuzzy model reference control is calculated by the weight average method, given inputs, $\dot{P}_{n,robot}(k)$ the final output is the weight average of $u_{neuro}(k)$ as shown in equation (19).

$$u_{neuro}(k) = \frac{\sum_{j=1}^{m} \mu_j(k) w_j(k)}{\sum_{j=1}^{m} \mu_j(k)} \tag{19}$$

where

$$\mu_j(k) = A_1^i(\dot{P}_{n,robot}(k)) \tag{20}$$

where $A_1^i(\dot{P}_{robot}(k))$ is calculated as in equation (8).

The adaptive neuro-fuzzy model reference control learns to track the desired velocity reference model, $r(k)$. The velocity reference model is defined as the function of the position error as the following.

$$r(k) = f(P_{robot}(k)) \tag{21}$$

where $f(\bullet)$ is an linear or nonlinear function.

The weights of the adaptive neuro-fuzzy model reference control are modified with the steepest gradient method by trying to minimize a cost function. The cost function is defined as the square of the difference between the velocity reference model and the actual velocity as expressed by the following.

$$E = \frac{1}{2}(r(k) - \dot{P}_{robot}(k))^2 \tag{22}$$

The equation for updating the weight is described as the following.

$$w_j(k+1) = w_j(k) - \eta \frac{\partial E}{\partial w_j} \tag{23}$$

where $\eta \geq 0$ is the learning rate.

By using the chain rule, the adjusted weights can be expressed and calculated as the following.

$$\frac{\partial E}{\partial w_j} = \frac{\mu_j(k)}{\sum_{j=1}^{m} \mu_j(k)}(r(k) - \dot{P}_{robot}(k)) \tag{24}$$

4. Simulation and Experimental Results

4.1 Simulations of yaw control

In the simulation study, the neuro-fuzzy yaw control is evaluated. The objectives of the study are to verify the desired procedure and evaluate the control performance of the neuro-fuzzy control. The yaw dynamic mathematical model of a flying robot is taken from [4]. The discrete-time LTI model is given by the followings.

$$x(k+1) = Ax(k) + Bu(k) \tag{25}$$

$$y(k) = Cx(k) \tag{26}$$

where

$$A = \begin{bmatrix} 1 & T_s \\ -0.1376 & 0.8947 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ -2.0269 \end{bmatrix} \text{ and } C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

where T_s is a sampling time, that is 0.02 s.

In order to generate the training data, the conventional proportional control is used to simulate the pilot control signal. The training data should have at lease two regions of overshoot. To accomplish this propose, the proportional gain of the proportional controller is tuned until the oscillation occurs. Fig. 7 shows the step input response of the yaw control with the proportional control. The yaw command is 45 degrees and the proportional gain is 0.98.

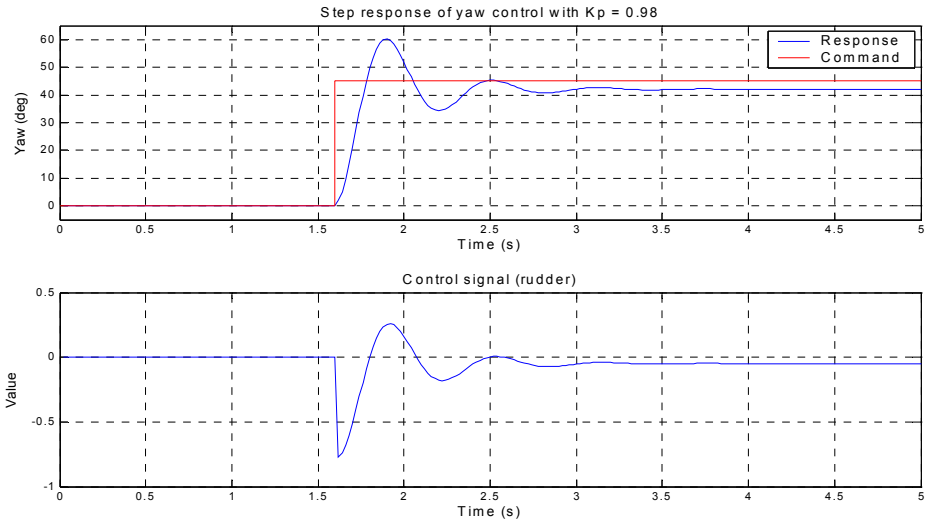


Fig. 7. Step input response of yaw control

The training region is selected. The yaw error and change of yaw error are generated. Fig. 8 shows the training data and the training result for the neuro-fuzzy yaw control.

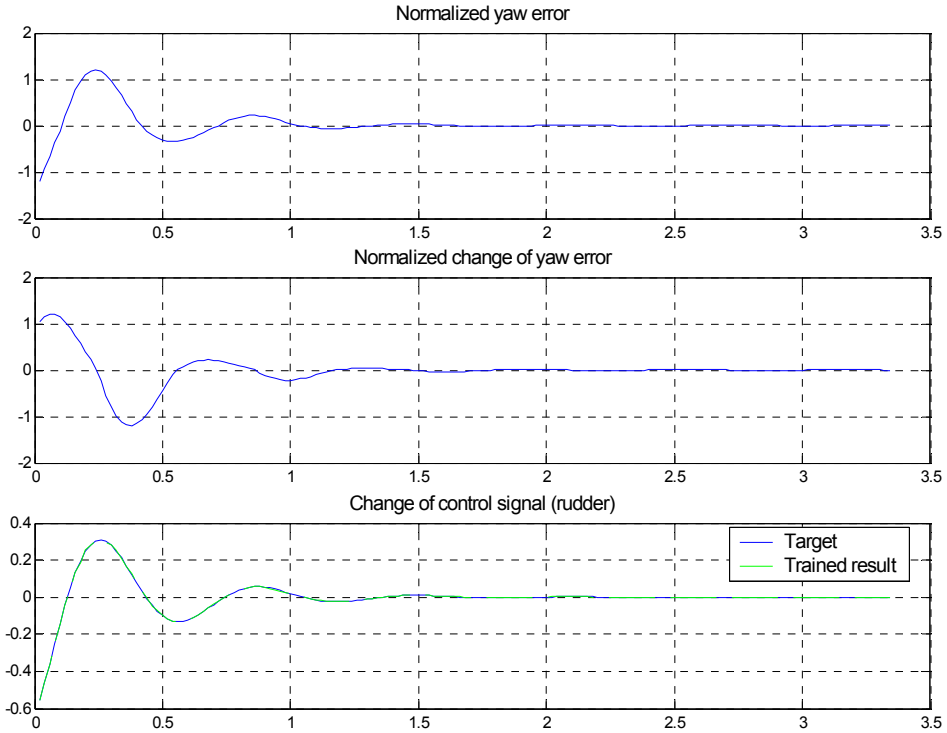


Fig. 8. Inputs and target data for neuro-fuzzy yaw control

In the simulation, there are 7 membership functions for each input. Each linguistic value is expressed by its mnemonic; for example, *NB* stands for "negative big", *NM* stands for "negative medium", *NS* stands for "negative small", *ZO* stands for "zero", and likewise for the positive (*P*) mnemonic. The membership functions are shown in Fig. 9. The yaw

error and change of yaw error are normalized to the range between –1.2 and 1.2. The normalized factors are shown in Table 1.

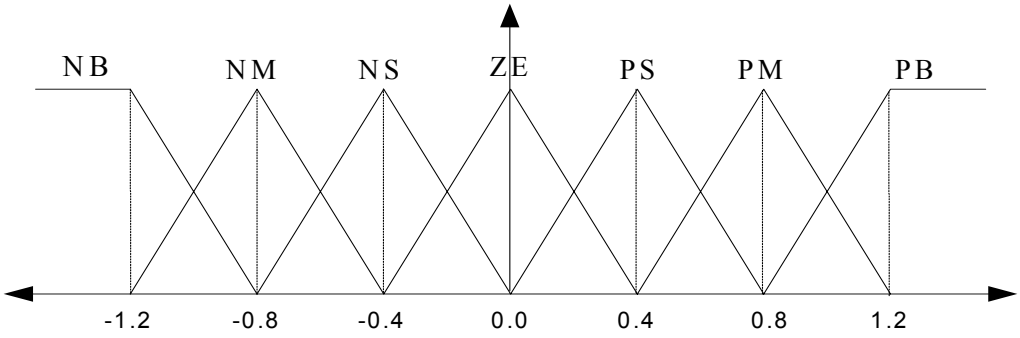


Fig. 9. Inputs and target data for neuro-fuzzy controller

Fig. 10 shows the simulation result of the neuro-fuzzy yaw control. The blue response shows the oscillation of the output response. It is indicated that the neuro-fuzzy need to be re-tuned. To damp out the oscillation, the change of yaw error gain, $k_{\Delta e}$, need to be increased. The control performance is improved as shown by the green response and yellow response by increasing of the change of yaw error gain, $k_{\Delta e}$, to 2.0 and 5.0 respectively. With the gain, $k_{\Delta e}$, of 5.0, there is a steady state error. The steady state error is eliminated by locally fine-tuned the neuro-fuzzy controller. The fine-tuning is done online as the result in the cyan response. The fine-tuning uses the steepest gradient method as in equation (12). The weights of the neuro-fuzzy controller are tuned locally by using the following condition.

$$\eta > 0.0 \qquad ,if \qquad |e(k)| \le a \ and \ |e(k)| \ge b \ and \ |\Delta e(k)| \le c$$

$$\eta = 0.0 \qquad ,otherwise$$

where a , b and c are positive constants.

In the simulation, the learning rate is 0.02. The values of a , b and c are 4.0, 0.05 and 0.1, respectively. The constant a and c are used to prevent the online fine-tuning not to modify the global structure of the neuro-fuzzy controller. The constant b is the threshold of the tuning.

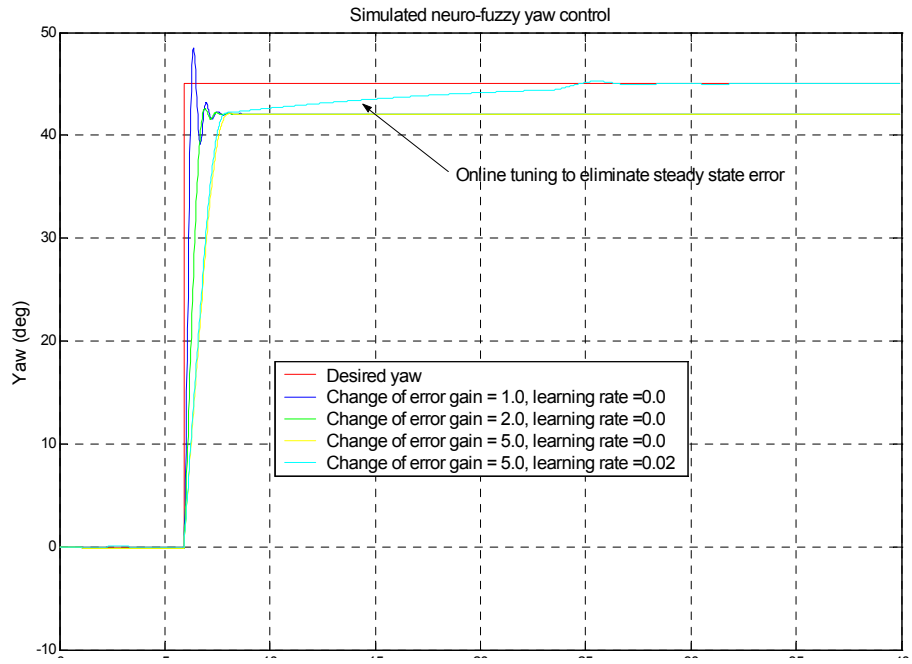


Fig. 10. The output response of neuro-fuzzy control

Table 1. Neuro-fuzzy yaw control parameters, * indicated for the values after re-tune the controller

g_1		g_2		k		
g_{1neg}	g_{1pos}	g_{2neg}	g_{2pos}	k_e	$k_{\Delta e}$	k_δ
2.5453	3.8146	25.7061	10.9662	1.0 *1.0	1.0 *5.0	1.0 *1.0

4.2 Experiments on neuro-fuzzy yaw control

In this section, the experiment of the yaw control is presented. The training data are generated by applying the open-loop stimulus control signal to the yaw axis, while try to maintain the flying robot in trim in the others axes. The signal causes the robot to oscillate about the z-axis. The yaw and the pilot control signals are recorded. Fig. 11 shows the recorded data.

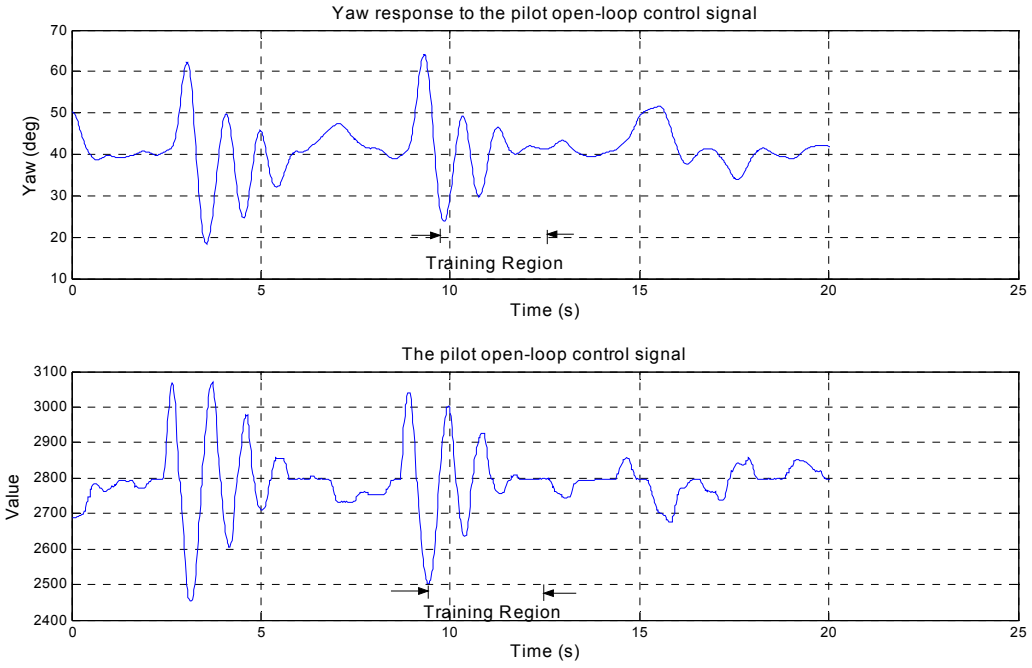


Fig. 11. Recorded yaw and the rudder signal

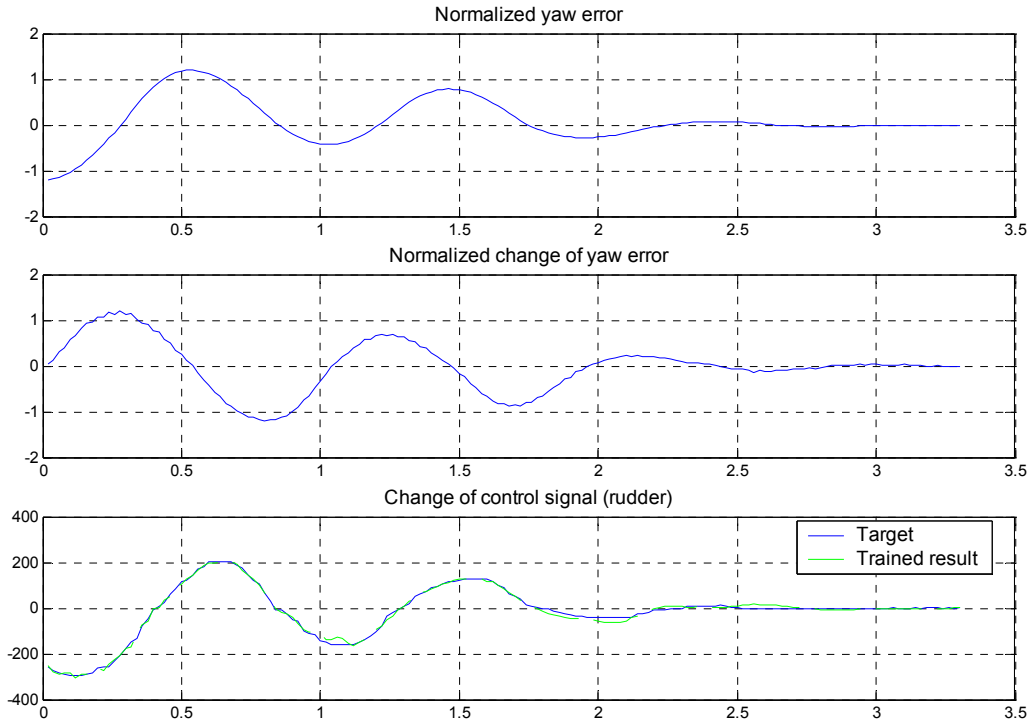


Fig. 12. Training data for neuro-fuzzy yaw control

Fig. 12 shows the training data and the offline training result of the neuro-fuzzy yaw control. The membership functions are same as used in the simulation. The desired parameters are shown in Table 2.

Table 2. Neuro-fuzzy yaw control parameters, * indicated for the values after re-tune the controller

g_1		g_2		k		
g_{1neg}	g_{1pos}	g_{2neg}	g_{2pos}	k_e	$k_{\Delta e}$	k_δ
0.0529	0.0684	0.7619	0.4706	1.0 *1.0	1.0 *2.1	1.0 *1.39

After the offline training, the neuro-fuzzy is re-tuned. The result is shown in Fig. 13. At the beginning, the gains of the neuro-fuzzy control are manually re-tuned, until the acceptable control performance is achieved. After that, the neuro-fuzzy control is fine-tuned online. This fine-tuning process eliminates the steady state error.

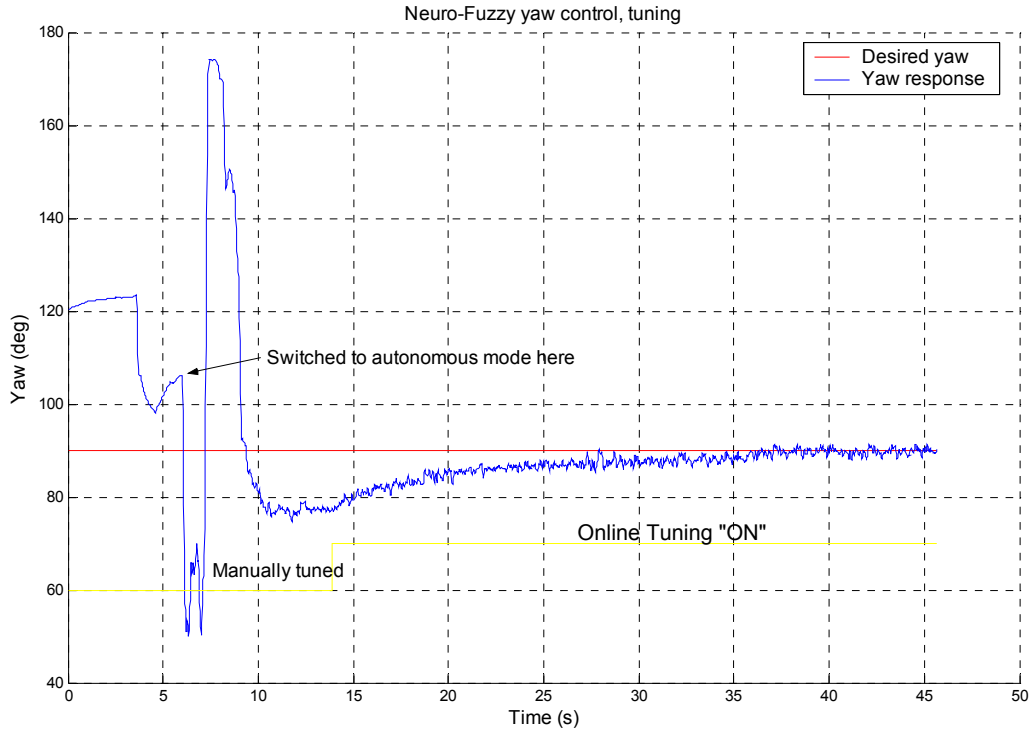


Fig. 13. Tuning result of neuro-fuzzy yaw control

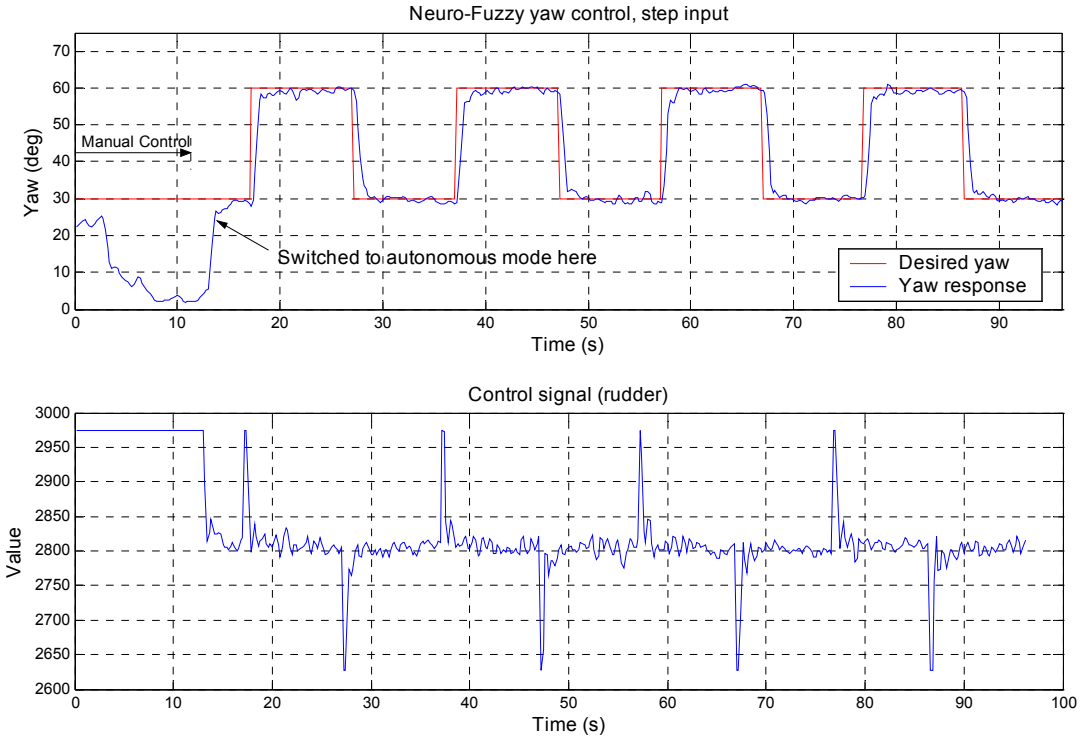


Fig. 14. Step input response of neuro-fuzzy yaw control

The final yaw control experiment is shown in Fig. 14. The step inputs are applied. According to the proposed control algorithm, the steady state error as well as the smooth response is achieved as seen in Fig. 14. There are not the overshoot/oscillations. The steady state error is always zero.

4.3 Experiments on position control with Hybrid-ANFMRC

Fig. 17, Fig. 18 and Fig. 19 show the experiment results of the lateral position, longitudinal position and altitude control, respectively. The outputs of the lateral position, longitudinal position and altitude control are the desired roll, desired pitch and the change of collective command, respectively. The roll and pitch control are designed according to the yaw control in section 4.2. The proportional gains of the lateral and longitudinal position control are 8.0. The proportional gain of the altitude control is 30.0. The lateral and longitudinal position command are 0 meter. The altitude command is 13.0 meter. The learning rate values for the lateral, longitudinal position and altitude control are 0.4. The velocity reference model is defined as the linear function of the position error as shown in Fig.15. The membership functions are designed by using 7 symmetrical triangle functions as shown in Fig. 16. The robot velocities are normalized within the range of –1.2 and 1.2. There are 7 elements of the weight for each controller, which are initialized to zero at the beginning.

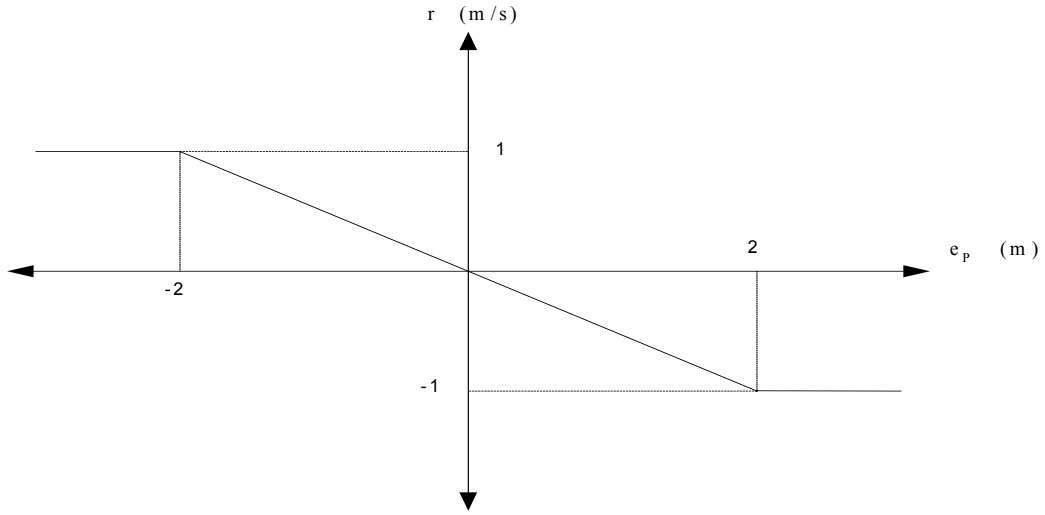


Fig. 15 The velocity reference model

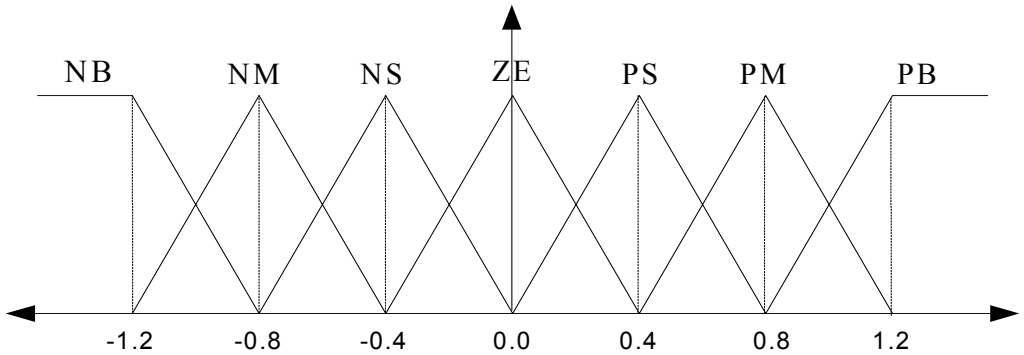


Fig. 16 Membership function for Hybrid-ANFMRC

In Fig. 17 and Fig. 18, applying only the proportional control at the beginning. After that, the learning process of the Hybrid-ANFMRC is activated, which is indicated by the yellow line in each figure. After the learning is started, the controller adapts the control parameters and learns how to control the position of robot. Finally, it can track the desired position with zero steady state error.

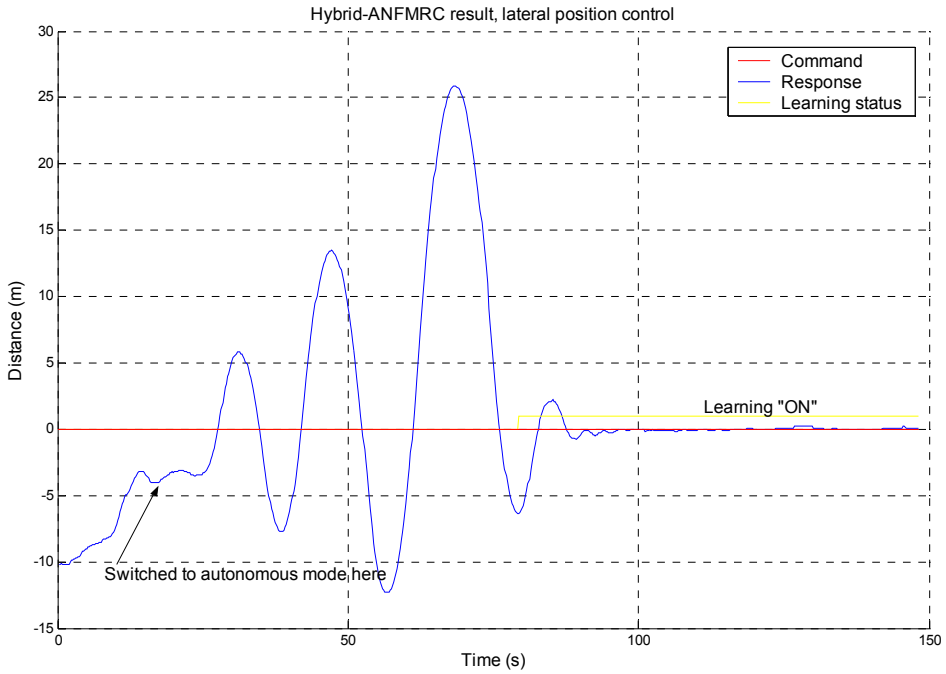


Fig. 17. Hybrid-ANFMRC, lateral position

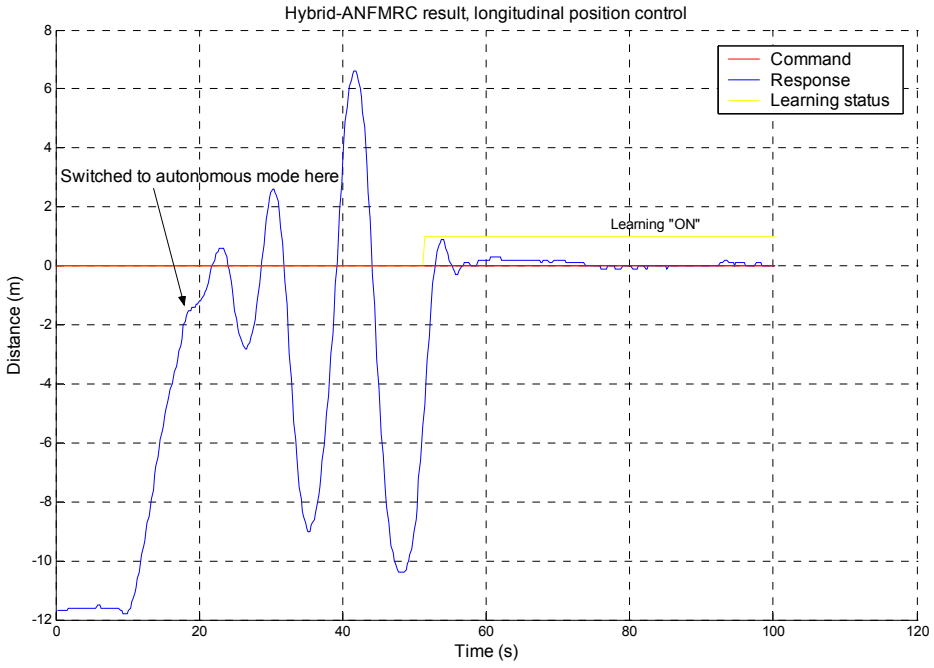


Fig. 18. Hybrid-ANFMRC, longitudinal position

In Fig. 19, the control is switched between the pilot controlled and the computer controlled. Every time the pilot takes control the robot, the learning process is stopped. In the first round, there are the large oscillations. The next round, the oscillation are reduced. Finally, the oscillations are eliminated. The controller can learn to control the altitude of the robot effectively.

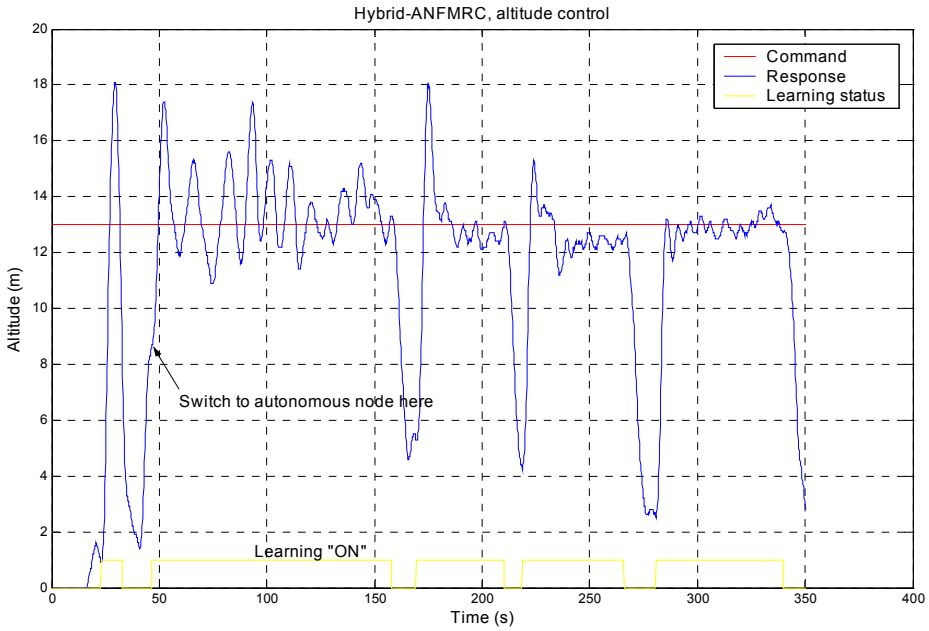


Fig. 19. Hybrid-ANFMRC, altitude

From the results, it was verified that the proposed Hybrid-ANFMRC was very effectively to control position of the flying robot. The weights of each controller are shown in Table 3.

Table 3. Weights of Hybrid-ANFMRC

		w_1	w_2	w_3	w_4	w_5	w_6	w_7
Lateral	Before	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	After	104.9	13.65	10.67	-7.48	-14.31	-15.64	-58.63
Longitudinal	Before	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	After	-140.77	-15.8	-15.14	-2.18	8.35	13.61	83.79
Altitude	Before	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	After	-92.7	-23.8	-59.98	53.19	95.2	39.16	119.6

4.4 Robustness of Hybrid-ANFMRC

In this section, the robustness of the hybrid adaptive neuro-fuzzy model reference is presented. In order to evaluate the robust performance of the proposed control algorithm, the longitudinal position control is studied. The proportional gain of the longitudinal position controller is varied. In Fig. 20, the proportional gain of 2.0 is used. In Fig. 21 and Fig. 22 the proportional gain of 4.0 and 8.0 are used, respectively.

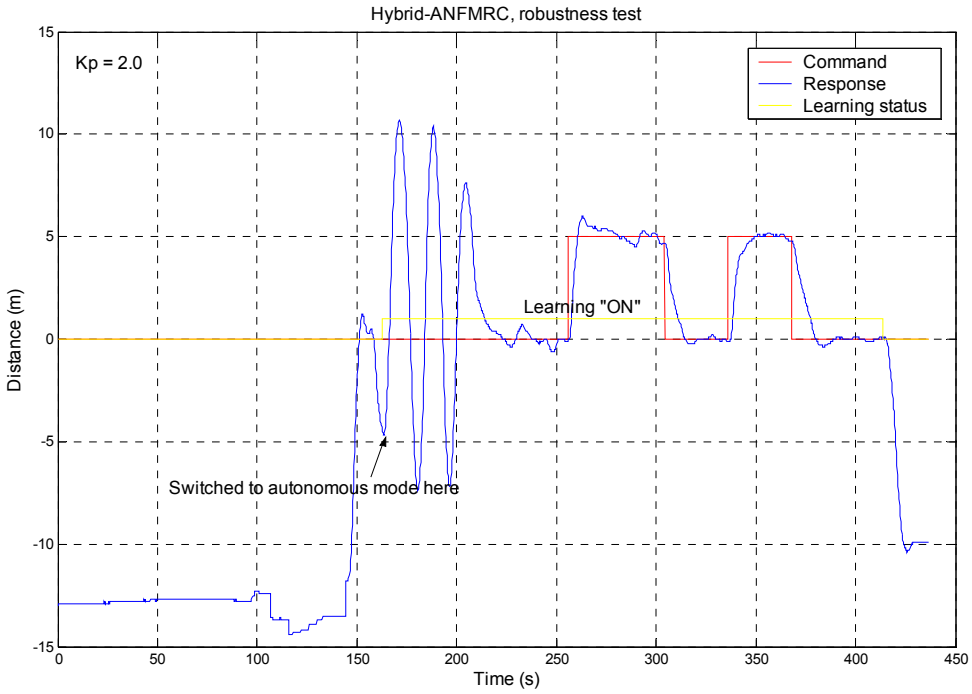


Fig. 20 Response of Hybrid-ANFMRC with $k_P = 2.0$

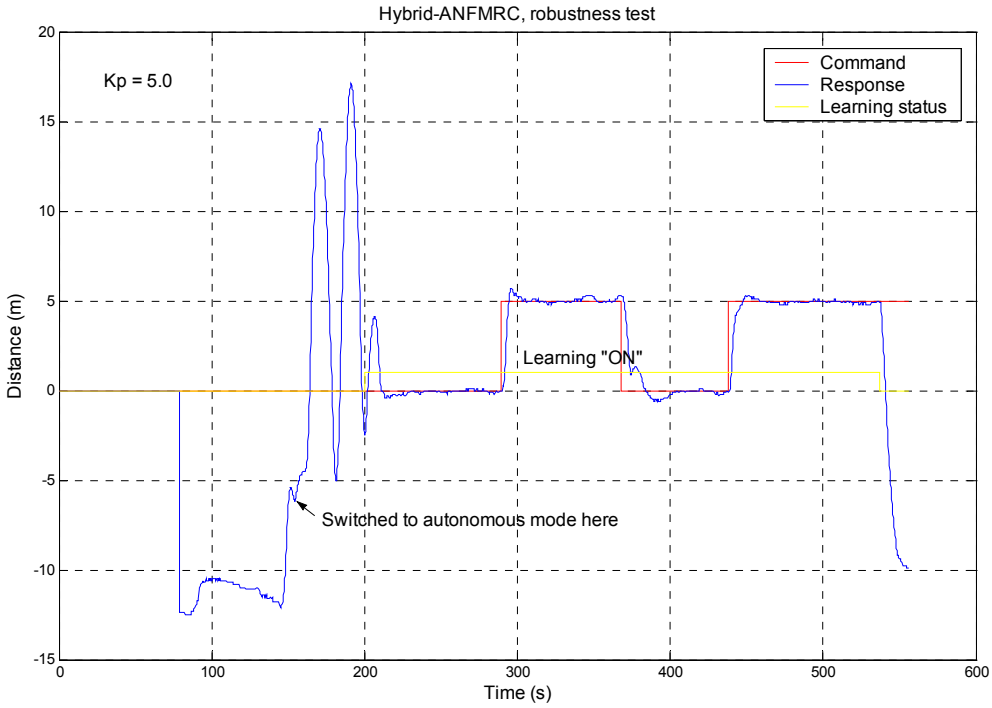


Fig. 21 Response of Hybrid-ANFMRC with $k_P = 4.0$

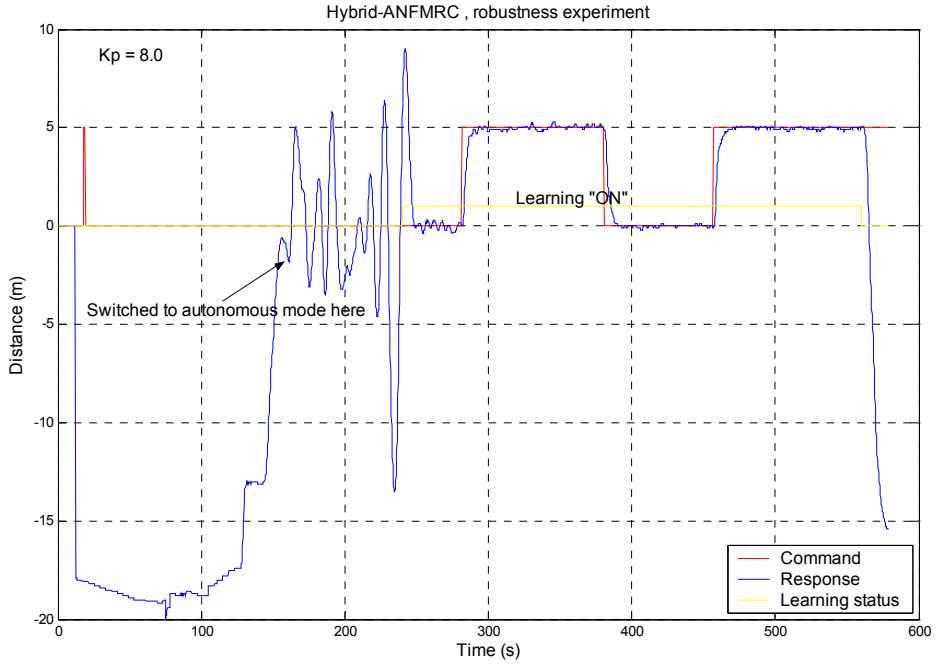


Fig. 22 Response of Hybrid-ANFMRC with k_P = 8.0

In Fig. 20, Fig. 21 and Fig. 22, the results indicate that the Hybrid-ANFMRC robust to the variation of the proportional gain. The control can learns and adapts itself to control the position of the flying robot. In the next section, the experiment on fully autonomous flight will be presented.

4.5 Experiments on fully 6-DOF autonomous controls

In order to test the control performance of the proposed control algorithm, the flying robot is commanded to fly in a 10 meters by 10 meters square area. The flying robot is automatically controlled in 6 DOF, including roll, pitch, yaw, lateral position, longitudinal position and altitude. The altitude command is 13.0 meters above the ground. The position commands are changed sequentially among the 4 marked points. When the flying robot reaches the desired position within the radius of 0.30 meter, the position commands are changed to the next points. For simplicity, the yaw is maintained at 0 degree. In this experiment, all the weights of the control are initialized to zero at the beginning.

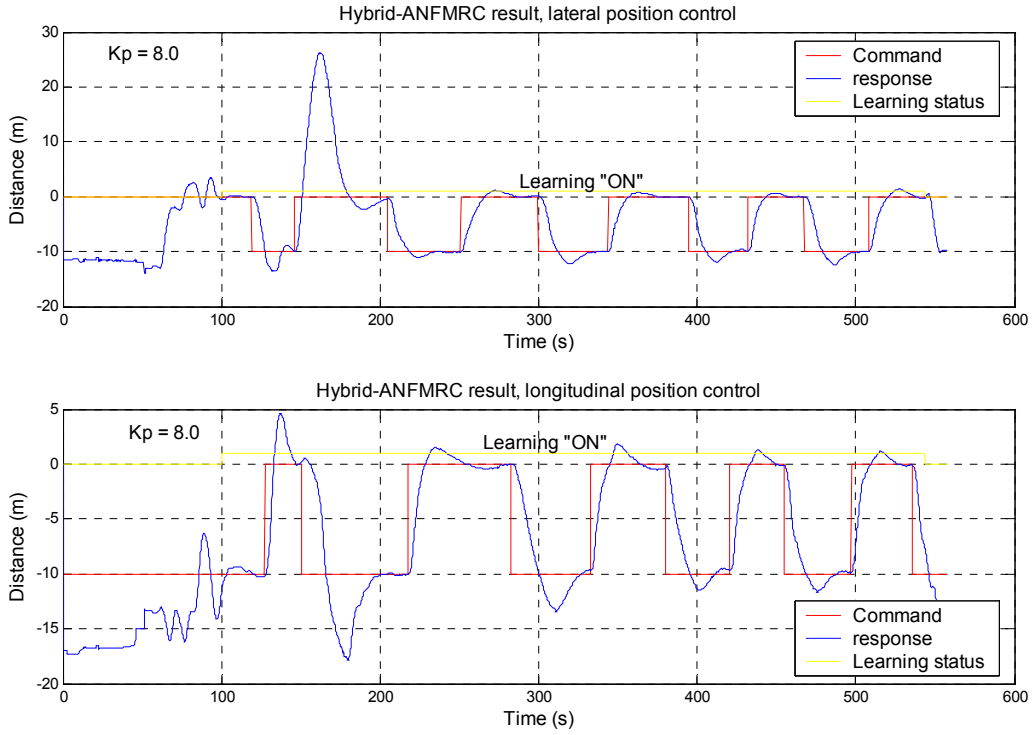


Fig. 23 Fully autonomous flight experiment results, lateral and longitudinal positions

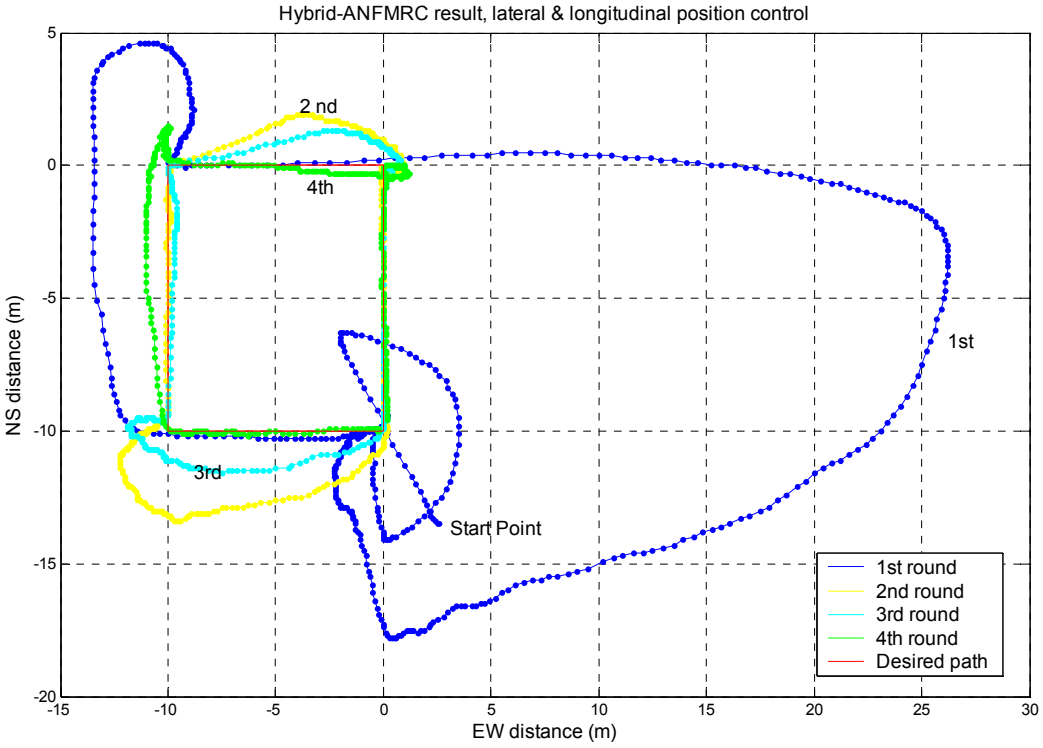


Fig. 24 Fully autonomous flight experiment results, top view

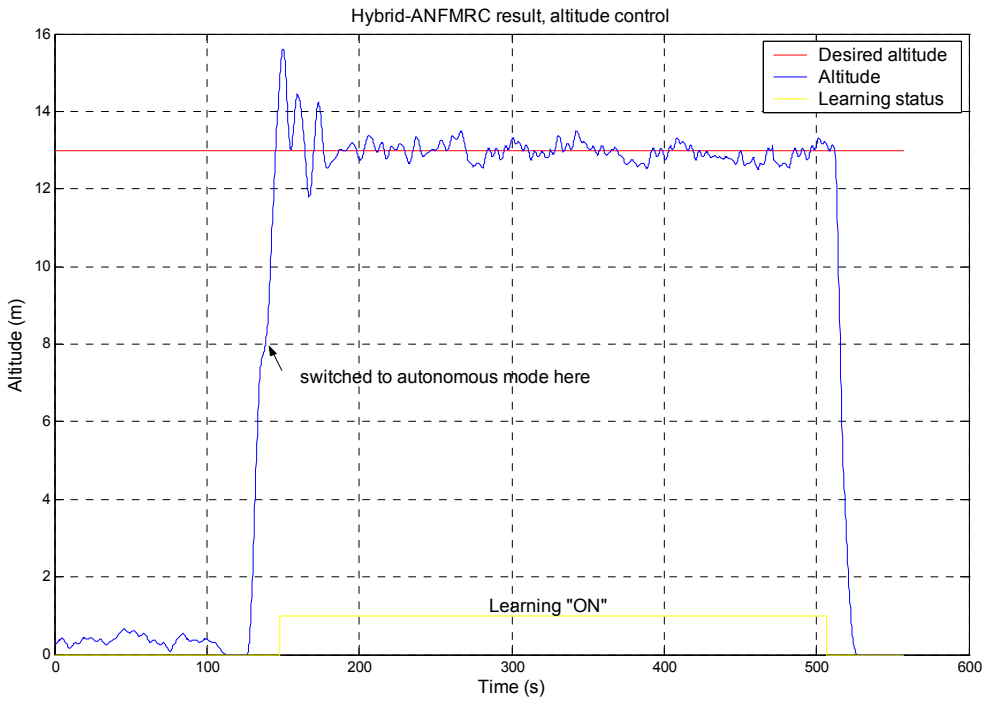


Fig. 25 Fully autonomous flight experiment results, altitude

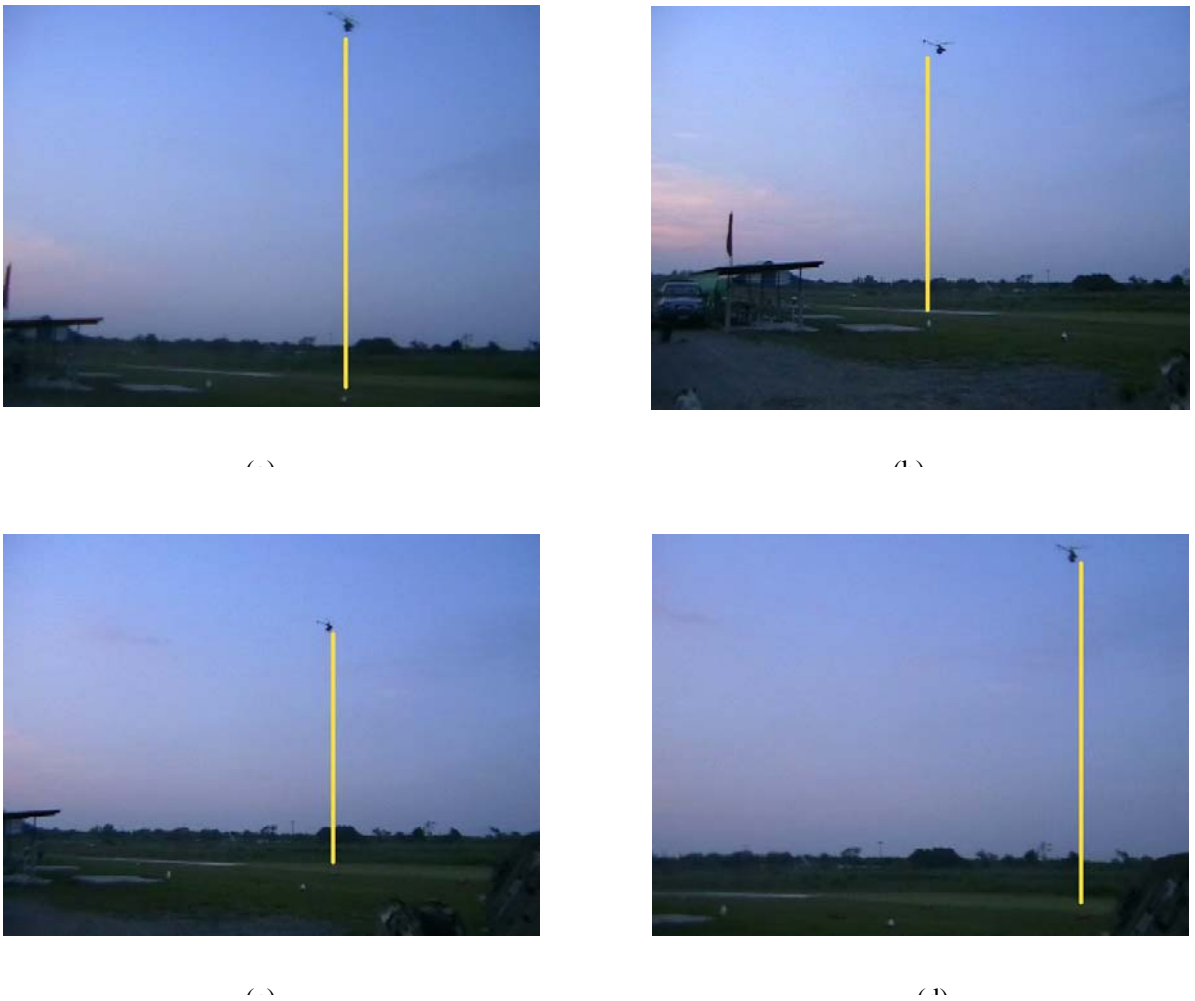


Fig. 26 Fully autonomous flight experiment result, pictures
(a) over the 1st point, (b) over the 2nd point, (c) over the 3rd point and (d) over the 4th point

In Fig. 23, Fig. 24 and Fig. 25, the first round of an autonomous flight shows the large overshoot. In the first round, the weights of the controller are all set to zero that resulted in an overshoot. In the next rounds, the tracking performance is better. In the final round, the tracking performance is completely successful. The control starts to learn from scratch, until it can adapts itself to perform a good control performance.

5. Conclusion

In this paper, the neuro-fuzzy control and the Hybrid-ANFMRC are evaluated. The neuro-fuzzy is applied to control the roll, pitch and yaw of the flying robot. The control is trained using the flight data and re-tuned to achieve the desired response. After the roll, pitch and yaw controls are accomplished, the position control is evaluated. The control performance of the Hybrid-ANFMRC is verified by the results from many experiments. The proposed control algorithm shows the good performance even when the proportional gain is changed. The control can be designed without using the mathematical model of the plant. The experiments have shown that the proposed control algorithms are able to successfully control the flying robot both in hover and moving flight. With the proposed control algorithms, using the different velocity reference model can shape the system response. In this paper, the experiment used only the simple linear reference model. For a better response, the exponential or any nonlinear reference models are also useful.

6. Acknowledgements

This work is financially supported by Thailand Research Fund.

References

[1] C. Teng, C.S. George. Neural fuzzy systems. Prentice Hall Inc; 1999.
[2] Gordon Wyeth, Gregg Buskey, Jonathan Roberts. Flight control using an artificial neural network.. University of Queensland, 2002.
[3] Jame F. Montgomery and George A. Bekey. Learning helicopter control through "Teaching by Showing".
[4] J. Morris, M. van Nieuwstadt, and P. Bendotti. Identification and control of a model helicopter in hover. In Proceeding of the American Control Conference 1994.
[5] M. Sugeno. Development of an intelligent unmanned helicopter. Fuzzy modeling and control, selected works of M. Sugeno. P.13-43.
[6] Ming-Chang S., Niarn-Liarng L. Self-tuning neural fuzzy control the position of pneumatic cylinder under vertical load. National Cheng Kung University, Taiwan.
[7] S. Saripalli, J. M. Roberts, P. I. Corke, G. Buskey. A tale of two helicopters. Robotics research lab, University of Southern California, 2002